Architecture Principles Compliance Analysis

João Alves[1], André Vasconcelos[1,2] and Pedro Sousa[1]
[1]*Instituto Superior Técnico, University of Lisbon, Av. Rovisco Pais, Lisbon, Portugal*
[2]*INESC-Inovação, Instituto Superior Técnico, Lisbon, Portugal*

Keywords: Enterprise Architecture, Architecture Principles, Architecture Analysis, Architecture Principle Compliance Analysis.

Abstract: The architecture principles play a key role in the enterprise architecture evolution. However, the architecture does not always address the principles intentions, which could result in unplanned deviations. Through the related work is perceptible the nonexistence of an architecture analysis based on architecture principles. Hereupon, this research proposes an architecture analysis to evaluate the architecture compliance with architecture principles. The proposed analysis, based on ArchiMate consists in the principle formalization where the principle expected impact is recognized. This analysis enables to identify the principle compliant elements in an enterprise architecture description. This analysis has been applied in one of the largest Portuguese insurance companies to analyse the compliance of some specific architectures. The analysis feasibility presents this research as a contribution to the architecture principles field.

1 INTRODUCTION

Modern enterprises face a range of challenges imposed by their environment (Op't Land et al., 2008) which impacts how they hold their evolution, making them transform. This is where organizations position the enterprise architecture (EA) as an instrument to coordinate and steer their transformation (Greefhorst and Proper, 2011). The EA design defines the delivered services and all the alignment between the underlying business processes, information systems and IT infrastructure (Greefhorst and Proper, 2011). The robustness of this design is critical to face the imposed challenges.

Hereupon, the EA design must evolve in order to make effective the organization adaption to the environment. To properly guide this evolution, the architecture principles are positioned as the key ingredient. The architecture principles provide rules and guidelines to inform and support the way in which an organization sets about fulfilling its mission (The Open Group, 2009). Therefore, it's important that EA design complies with their guiding principles, which is not always achieved. This emphasizes the need for an EA compliance evaluation based on architecture principles.

However, the related work study shows that an EA analysis to evaluate the EA compliance with their guiding principles still lack. Hereupon, our vision pretends to formalize architecture principles, based on ArchiMate to enable their EA compliance analysis. This formalization enables to analyse an enterprise architecture description (EAD) through the detection of architecture structures that represent the principle expected impact and consequently identify their compliant elements.

This work is organized as follows. In section 2 are explained the principal domains related with this research and its motivation. In section 3 the approach behind the research proposal is presented. In section 4, the proposed analysis based on two principles is highlighted. In section 5 the research proposal feasibility is demonstrated in a real case study and finally the section 6 concludes the paper and provides research future directions.

2 RELATED WORK AND MOTIVATION

2.1 Enterprise Architecture

The EA can be defined as a coherent whole of principles, methods, and models that are used in the design and realisation of an enterprise's

Alves J., Vasconcelos A. and Sousa P..
Architecture Principles Compliance Analysis.
DOI: 10.5220/0004882803280334
In *Proceedings of the 16th International Conference on Enterprise Information Systems* (ICEIS-2014), pages 328-334
ISBN: 978-989-758-029-1

organisational structure, business processes, information systems, and infrastructure (Lankhorst, 2009). It enables a better decision making by sharing knowledge on architecture decisions and provides a way to describe and control an organization's structure, processes, applications, systems, and technology in an integrated way (Lankhorst, 2009).

The analysis intended by this research is intimately connected how EA could be represented. The ArchiMate comprises an EA modelling language providing precise descriptions of the architecture in different domains and different stakeholders, a feature that is not allowed in other modelling languages (The Open Group, 2012). The integrated representation between domains could turn easier to analyse the principle impact that propagates through multiple domains. The possibility to extend the ArchiMate metamodel (The Open Group, 2012) represents also another important issue. Some specific cases in principle analysis could require the unambiguously identification of a certain element or relationship that is not endorsed by the ArchiMate metamodel. These considerations justify the ArchiMate use in the proposed analysis.

2.2 Architecture Principles

The architecture principles can be seen as general rules and guidelines, intended to be enduring and seldom amended, that inform and support the way in which an organization sets about fulfilling its mission (The Open Group, 2009). They play a prominent role in the EA development giving advice how to design target architecture by restricting the design freedom of EA transformation projects (Aier et al., 2011). The architecture principles to be really effective and be considered good principles they must have a clear semantic, understandable syntax and the right focus (Lindström, 2006; Van Bommel et al., 2007). However, if any of these characteristics are violated some deviations in the expected impact could emerge (Greefhorst and Proper, 2011). This fact emphasises the need to verify if the EA impact is the prescribed by the architecture principles.

The principle application consists in the transformation activity, which is separated in two types. The first one called derivation consists in the principles transformation into statements that are relevant in a more specific context. The other is related with the principle transformation to models. This transformation it´s build on the fact that architecture principles can be the rationale behind a number of elements in the model and for their relationships (Greefhorst and Proper, 2011).

It's also important to relate the transformations with the respective compliance management. In the compliance management is advised the principle refinement into requirements and then in design decision to perform the compliance verification (Greefhorst and Proper, 2011). However, this advice maps with the derivation transformation. So, it's evident a lack of a compliance verification to the principle transformation to models and it is here that our work presents as a contribution.

Finally, the architecture principles used are selected from the catalogue in (Greefhorst and Proper, 2011). Another catalogue is provided by TOGAF (The Open Group, 2009) although the principles from the previous catalogue present some advantages. They are based on real-world architectures (Greefhorst and Proper, 2011), they are aligned with ArchiMate and are more level-specific (Vieira, 2012).

2.3 Architecture Analysis

The EA discipline advocates the use of models to support decision-making (Johnson et al., 2007). These decisions can be supported by appropriate analysis techniques that show why a solution is better or to detect inconsistencies (Šaša and Krisper, 2011). Lankhorst (2009) describes different architecture analysis techniques that can be used with ArchiMate. Quantitative and qualitative analysis techniques are distinguished.

Quantitative analysis focuses on the quantitative aspect of relationships between different EA elements and layers. It can be used for optimization by quantifying the effect of alternative design choices obtaining measures to support impact-of-change analysis (Šaša and Krisper, 2011). Qualitative analysis enables to understand how a system that conforms to the architecture works, to find the impact of a change on the architecture, or to validate the architecture correctness. This analysis distinguishes structural and dynamic aspects (Lankhorst, 2009). The structural analysis is used to determine the EA change impact which implies traverse the architecture and consider each relation and its meaning to determine whether the change might propagate. Description logics are useful formalisms to perform this analysis. For dynamic analysis, techniques based on formal interpretations are used. Dynamic analysis improves consistency and focuses on logical aspects of the models. (Šaša and Krisper, 2011)

Other approaches based on EA patterns exist for

business support analysis (Šaša and Krisper, 2011). The approach used consists in the pattern formalization to detect architecture structures that characterize each pattern. The detection of the referred structures enables to be aware of what could be changed and how the EA could evolve. The pattern formalization is based on ArchiMate.

Hereupon, the proposed analysis could be positioned in the structure analysis, however it is not intended to analyse the EA change impact. It allows evaluating the EA coherence which could result in the improvement of the architecture dynamics. These improvements could represent the rationale underlying the prescribed architecture principle.

3 APPROACH

The architecture principles are considered the rationale for the existence of several EA elements and relationships (Greefhorst and Proper, 2011), which could position EADs as artifacts that provide relevant information for the compliance analysis. The approach underlying the proposed analysis is based on (Šaša and Krisper, 2011). Initially it consists in the identification of the EA elements and relationships needed to perform the compliance analysis. Then, the architecture structures that represent the principle expected impact are recognized. This recognition enables to determine the compliant elements in the analyzed EAD. So, the used approach is composed as follows.

- **To Define Relevant EA Perspectives.** These perspectives are the viewpoints that represent the structures impacted by the principle. Their goal is to ensure that corresponding views illustrate exactly the relevant elements for the analysis.
- **To Define Characteristics That Address the Principle Perspectives**. These characteristics define the prescriptions imposed by the principle. They enable to recognize the principle expected impact in the EAD.

In summary, we represent an architecture principle as a set of elements, which is formalized with its membership conditions. If an EA element respects the principle membership conditions it is compliant.

4 PROPOSAL

To perform the proposed analysis the symbols used in the principle formalization are presented in Table 1. These symbols based on (Šaša and Krisper, 2011) represent the ArchiMate elements (The Open Group, 2012) impacted by the considered principles. However, it's important to notice that not every element belongs to the ArchiMate metamodel. The new elements and relationships correspond to extensions to the metamodel. The reason for each extension is explained in the principle analysis that requires it. (The Open Group, 2012).

Table 1: Symbols for principle formalization.

Symbols			
PIA	Set of all elements and relations of an EAD	*ACPreL*	Set of all presentation logic application components in the EAD: $ACPreL \subseteq AC$
BR	Set of all business roles in the EAD	*ACProL*	Set of all process logic application components in the EAD: $ACProL \subseteq AC$
CR	Set of all customers in the EAD: $CR \subseteq BR$	*ACBL*	Set of all business logic application components in the EAD: $ACBL \subseteq AC$
BI	Set of all business interfaces in the EAD	*ACDL*	Set of all data logic application components in the EAD: $ACDL \subseteq AC$
EF	Set of all electronic forms in the EAD: $EF \subseteq BI$	$(a,b) \in$ *Realization*	a is related to b with the Realization relationship : *a* realizes *b*
BS	Set of all business services in the EAD	$(a,b) \in$ *Composition*	a is related to b with the Composition relationship: *a* is composed of *b*
AS	Set of all application services in the EAD	$(a,b) \in$ *Aggregation*	a is related to b with the Aggregation relationship: a aggregates b
AC	Set of all application components in the EAD	$(a,b) \in$ *Used by*	a is related to b with the Used by relationship: *a* is used by *b*
DO	Set of all data objects in the EAD	$(a,b) \in$ *Provide*	a is related to b with the Provide relationship: *a* provides *b*
$(a,b) \in$ *Creation*	a is related to b with the Creation relationship: *a* creates *b*	$(a,b) \in$ *Assignment*	a is related to b with the Assignment relationship: *a* is assigned to *b*

Table 2: Applications are modular principle (Greefhorst and Proper, 2011).

A.28 Applications Are Modular (AAM)
Type of information: application **Quality attributes:** reliability, maintainability, portability
Rationale: • Modularized applications are much easier to develop, maintain, reuse and migrate than monolithical applications. • Modularized applications are also more reliable since changes have a more localized and therefore predictable impact.
Implications: • Applications are decomposed into components that have limited and acyclical dependencies on other components. • Application components are units of configuration management and deployment. • Application components have a logical and documented layered structure, where lower level layers are independent of higher level layers. • Presentation logic, process logic, business logic and data exist in separate layers or components.

It's also relevant to understand how the principle perspectives are defined. If *PIA* represents a set of all element and relationships of an EAD, then a viewpoint can be defined as a function *vp* that maps a given EA into a subset of its elements and their relations. Function *vp(PIA)=P*, *P⊆PIA*, where *P* represents a view of the EA from the viewpoint *vp*. Hereupon, two functions are defined to represent a viewpoint (Šaša and Krisper, 2011):

- Function *Elt(x)*, where *x⊆PIA*, is a function which returns all elements in a given EAD *x* or in a given view *x* of an EA.
- Function *Rel(x)*, where *x⊆PIA* is a function which returns all relationships in a given EAD *x* or in a given view *x* of an EA.

4.1 Applications Are Modular Analysis

The compliance analysis presented here is based on the principle highlighted in Table 2. Concerning this analysis, the needed ArchiMate extensions are represented as follows.

- The *ACPreL*, *ACProL*, *ACBL* and *ACDL* sets identify the application components that implement presentation, process, business and data logic, respectively.

4.1.1 Definition of AAM Perspectives

The AAM viewpoint (AAMV), *AAMV⊆PIA* is defined as follows.

$$(1)Elt(AAMV)=\{x|(x\in AC)\ \vee (x\in AS,\ \exists ac1,ac2\in AC:\ (ac1,x)\in Realization\ \wedge (x,ac2)\in Used\ by)\}$$

$$(2)Rel(AAMV)=\{(x,y)|x,y\in AC:\ (x.y)\in Composition\ \vee\ (x.y)\in Aggregation\}\ \cup\ \{(t,z)|z\in AC,t\in AS:\ (z,t)\in Realization\ \vee (t,z)\in Used\ by\}$$

4.1.2 Definition of AAM Characteristics

Concerning the AAM analysis, the principle prescribes that each monolithical application component only should implement one type of logic. The *MoAC1L* set identifies the application components that address this prescription.

$$(3)MoAC1L=MoAC\backslash(MoAC\cap ACSevL)$$

The *MoAC* set identifies the monolithical components and *ACSevL* defines the application components that implement at least two logics.

$$(4)MoAC=\{a|a\in AC\ \wedge (\nexists b\in AC:\ (a,b)\in Composition\ \vee (a,b)\in Aggregation)\}$$

$$(5)ACSevL=(ACPreL\cap ACProL)\cup(ACPreL\cap ACBL)\cup(ACPreL\cap ACDL)\cup(ACProL\cap ACBL)\cup(ACProL\cap ACDL)\cup(ACBL\cap ACDL)$$

The principle also prescribes about the dependency between application components. This dependency is related to the usage that a component makes from other component. To control this dependency the application components that correspond to a layer with lower abstraction level are independent of higher level components. So, the components with a certain type of logic only could use application components with lower or equal abstraction level logic. To verify this prescription the different logics have to be classified depending on the abstraction level. From the higher level to the lower, the presentation logic is followed by the process, business and data logic. The *ACU* set identifies the monolithical components that address the dependency prescription. The *ACPreLU*, *ACProLU*, *ACBLU* and *ACDLU* (Appendix) sets define for each logic the components that endorse the dependency prescription.

Table 3: Data are provided by the source principle (Greefhorst and Proper, 2011).

A.14 Data Are Provided by the Source (DPS)
Type of information: data, application **Quality attributes:** reliability, efficiency
Rationale: • When those who have the data also provide them, unnecessary intermediate layers (e.g. people or IT components) are prevented. • The performance and reliability of the data also increases, since each link in the chain adds performance overhead and potential errors.
Implications: • Electronic forms are provided to customers to enter their requests. • Applications acquire data from the source application.

$$(6) ACU = ACPreLU \cup ACProLU \cup ACBLU \cup ACDLU$$

The principle also prescribes about the application component composition. The components only should be composed by components in accordance with the prescriptions previously analysed. This guarantees the application layered structure intended by the principle. So, the *ACComp* set identifies the components that respect this prescription.

$$(7) ACComp = \{ac | ac \in (AC \setminus MoAC) \wedge (AppComp(ac) \subseteq ACU)\}$$

The *AppComp(ac)* function previously used identifies the application components that compose a certain component.

$$(8) AppComp(x) = \{ac | ac \in AC \wedge ((x,ac) \in Composition \vee (x,ac) \in Aggregation)\}$$

Hereupon, the principle compliant application components are identified by the AAM set.

$$(9) AAM = ACU \cup ACComp$$

4.2 Data Are Provided by the Source Analysis

The compliance analysis presented here is based on the principle highlighted in Table 3. Concerning this analysis the ArchiMate extensions needed are:

- The *Creation* and *Provide* relationships. They are created to avoid the access relationship ambiguity.
- The *CR* set identifies business roles that correspond to customers.
- The *EF* set identify business interfaces that represent electronic forms.

4.2.1 Definition of DPS Perspectives

The DPS viewpoint (DPSV), $DPSV \subseteq PIA$ is characterized as follows.

$$(10) Elt(DPSV) = \{x | (x \in AS) \vee (x \in AC, \exists a \in AS: (x,a) \in Realization) \vee (x \in DO, \exists b \in AS: (b,x) \in Provide \vee (b,x) \in Creation)\} \cup \{y | (y \in BS) \vee (y \in BI, \exists c \in BS: (x,c) \in Assignment \wedge \exists c \in BR: (x,c) \in Used\ by)\}$$

$$(11) Rel(DPSV) = \{(x,y,z) | x \in AS, y \in DO, z \in AC, ((x,y) \in Provide \vee (x,y) \in Creation) \wedge (z,x) \in Realization\} \cup \{(t,u,v) | t \in BR, u \in BI, v \in BS, (u,z) \in Assignment \wedge (u,t) \in Used\ by\}$$

4.2.2 Definition of DPS Characteristics

The principle prescribes that data object should be provided by its application source. The application source of a certain data is the application component responsible for its creation. So, the *ASDOAS* set identifies the application services that only provide data created by the component that realizes that services.

$$(12) ASDOAS = \{a | a \in AS \wedge (\exists do \in DO: (a,do) \in Provide) \wedge (\exists b \in ASAC(a): (b,do) \in Creation)\}$$

The *ASAC(a)* function identifies all application services realized by the application component that realizes a certain application service.

$$(13) ASAC(a) = \{as \mid as \in AS \wedge \exists ac \in AC: (ac,a) \in Realization \wedge (ac,as) \in Realization\}$$

It's also prescribed that should be provided electronic forms to customers enter their requests and use the delivered business services. This prescription is endorsed by the *BSEF* set.

$$(14) BSEF = \{bs | bs \in BS \wedge \exists bi \in EF: (bi,bs) \in Assignment \wedge \exists br \in CR: (bi,br) \in Used\ by\}$$

Hereupon, the *DPS* set identifies the principle compliant elements.

$$(15) DPS = ASDOAS \cup BSEF$$

5 CASE STUDY

To demonstrate our proposal the compliance analysis was applied to real EADs provided by one of the largest Portuguese insurance companies. The used architectures should address the considered principles and for that reason are used to perform this analysis. So, for each principle analysis several views describe the analysed architecture, where are marked in green the compliant elements resulting from the analysis application. It's important to notice that these views are based on the principle perspectives and their element names are letters due to confidentiality reasons. In Figure 1 and 2 each partition represents a view where the responsible set for the compliance analysis is also referred.

5.1 AAM Analysis Application

The AAM analysis is illustrated in Figure 1.

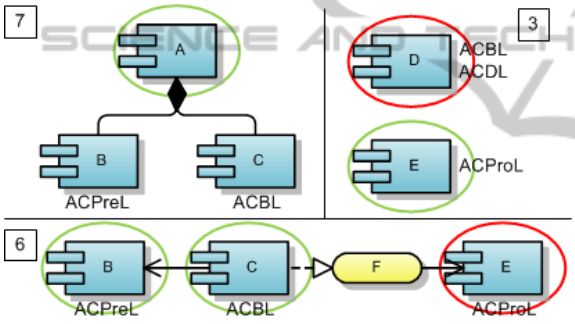

Figure 1: AAM compliance analysis.

5.2 DPS Analysis Application

The DPS analysis is illustrated in Figure 2.

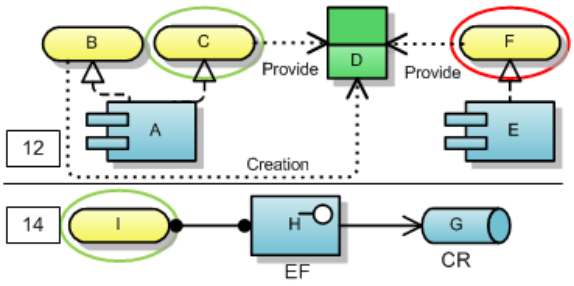

Figure 2: DPS compliance analysis.

5.3 Results Analysis

The recognition of the EA compliant elements through the proposal application also enables to acquire the knowledge of what the non-compliant elements are. Regarding the AAM analysis (Figure 1) the non-recognition of several elements as compliant is due to their development have been carried before the emergence of software application modularity principles. This past legacy represents the reason for the non-recognition of "D" and "E" as compliant elements.

Concerning the DPS analysis (Figure 2), the "F" non-recognition is explained by a redundancy in the insurance company middleware layer. The existence of this redundancy is known by the architects and is explained by the acquisition of the "A" and "E" components as a package. When other components want to obtain the "D" object from services provided by "A" they can't, since "E" is responsible for the communication with "A". So, "D" only could be provided through "E" which does not represent its application source. Consequently "F" is not recognized as compliant.

Hereupon, the proposed compliance analysis enable to evaluate the EA based on architecture principles. This analysis allows identifying the compliant elements through the principle expected impact.

6 CONCLUSIONS

In this paper was proposed an EA analysis based on architecture principles. This analysis enables to identify the compliant elements of an EAD with their guiding principles. The analysis endorses two different principles sufficiently objective to realize which impact they have on the EA. The principle expected impact and the ArchiMate language provide the basis for the approach underlying the proposed analysis. Initially are defined the architecture perspectives which provide the elements and relationships impacted by the principle. Then, for each perspective are formally defined the conditions prescribed by the selected principle. This formalization is used to verify if the elements of a certain perspective are or not compliant with the respective principle.

The analysis feasibility was demonstrated in real architectures where compliant elements are identified and the non-conformities are justified. As part of this research, this analysis will be extended to endorse other architecture principles and also it's expected their implementation in an EA management tool to a larger and complex analysis could be performed. To conclude, as identified in this work an EA analysis based on architecture principles still lack. Hereupon, our research

contributes to surpass this gap providing a mechanism that allows analysing the principle compliance through an EAD.

REFERENCES

Aier, S., Fischer, C., Winter, R., 2011. Construction and Evaluation of a Meta-Model for Enterprise Architecture Design Principles. Wirtschaftsinformatik Proceedings.

Greefhorst, D.,Proper, E., 2011. *Architecture Principles: The Cornerstones of Enterprise Architecture*, Springer, Berlin.

Johnson, P., Lagerström, R., Närman, P., Simonsson, M., 2007. Enterprise architecture analysis with extended influence diagrams. Information Systems Frontiers.

Lankhorst, M., 2009. *Enterprise Architecture at Work: Modelling Communication and Analysis*, Springer. Berlin, 2nd edition.

Lindström, A., 2006. On the syntax and semantics of architectural principles. In *HICSS '06, 39th Hawaii international conference on system sciences.* Computer Society Press.

Op't Land, M., Proper, H.A., Waage, M., Cloo, J., Steghuis, C., 2008. *Enterprise architecture—creating value by informed governance*, Springer. Berlin.

Šaša, A., Krisper, M., 2011. Enterprise architecture patterns for business process support analysis. *In Journal of Systems and Software*. Elsevier.

The Open Group, 2012. *ArchiMate 2.0 Specification*, Van Haren Publishing.

The Open Group, 2009. *TOGAF Version 9 TOGAF Series*, Van Haren Publishing.

Van Bommel, P., Buitenhuis, P.G., Stijn, J.B., Hoppenbrouwers, A., Proper, E.H.A., 2007. Architecture Principles – A Regulative Perspective on Enterprise Architecture. In Reichert M, Strecker S, Turowski K (eds) Enterprise modelling and information systems architectures Gesellschaft fuer Informatik.

Vieira, T., 2012. Evaluating Enterprise Architectures: From Principles to Metrics. MscThesis.

APPENDIX

$$(16)ACPreLU=\{ac1|ac1\in(ACPreL\cap MoAC1L)\wedge((\exists ac2\in MoAC1L:(ac2,ac1)\in Used\ by\vee(\exists as1\in AS:(as1,ac1)\in Used\ by\wedge(ac2,as1)\in Realization))\vee(\nexists ac3\in AC:(ac3,ac1)\in Used\ by\wedge\nexists as2\in AS:(as2,ac1)\in Used\ by))\}$$

$$(17)ACProLU=\{ac1|ac1\in(ACProL\cap MoAC1L)\wedge((\exists ac2\in(ACProL\cup ACBL\cup ACDL)\cap MoAC1L):(ac2,ac1)\in Used\ by\vee(\exists as1\in AS:(as1,ac1)\in Used\ by\wedge(ac2,as1)\in Realization))\vee(\nexists ac3\in AC:(ac3,ac1)\in Used\ by\wedge\nexists as2\in AS:(as2,ac1)\in Used\ by))\}$$

$$(18)ACBLU=\{ac1|ac1\in(ACBL\cap MoAC1L)\wedge((\exists ac2\in((ACBL\cup ACDL)\cap MoAC1L):(ac2,ac1)\in Used\ by\vee(\exists as1\in AS:(as1,ac1)\in Used\ by\wedge(ac2,as1)\in Realization))\vee(\nexists ac3\in AC:(ac3,ac1)\in Used\ by\wedge\nexists as2\in AS:(as2,ac1)\in Used\ by))\}$$

$$(19)ACDLU=\{ac1|ac1\in(ACDL\cap MoAC1L)\wedge((\exists ac2\in(ACDL\cap MoAC1L):(ac2,ac1)\in Used\ by\vee(\exists as1\in AS:(as1,ac1)\in Used\ by\wedge(ac2,as1)\in Realization))\vee(\nexists ac3\in AC:(ac3,ac1)\in Used\ by\wedge\nexists as2\in AS:(as2,ac1)\in Used\ by))\}$$