Using Gap Analysis to Support Feedback Loops for Enterprise Architecture Management

Philipp Diefenthaler

Softplant GmbH, 80992 Munich, E-Mail: Philipp.Diefenthaler@softplant.de

Dr. Bernhard Bauer

Universität Augsburg, Software Methodologies for Distributed Systems, 86135 Augsburg, E-Mail: Bernhard.Bauer@informatik.uni-augsburg.de

Abstract

Enterprise architecture models are created to support analysis and documentation in different phases of the architecture process. Gap analysis is used in many enterprise architecture processes as a means for planning architectural changes. Here we are using the gap analysis in the check phase to detect and quantify gaps between an actually achieved new as-is, the old as-is and the previously planned to-be. We introduce seven disjoint subsets, which are intersections and unions of the three original sets. These subsets provide necessary information to an enterprise architect to support the creation of feedback loops for enterprise architecture management.

1 Introduction

In this paper, we suggest a mechanism to check for the realization of planned changes as part of Enterprise Architecture Management (EAM). EAM as a management discipline integrates with the Deming cycle [6], which consists of four phases: plan, do, check and act.

In the plan phase, the enterprise architect creates architecture models for the current (as-is) and the desired (to-be) state and develops a plan how to realize the to-be state. This phase is supported by a number of techniques that use different approaches to determine necessary changes. Among others, gap analysis is used to compare the as-is and the to-be model ([8], [16]).

The do phase is concerned with implementing the change through programs and projects. Afterwards, it must be checked whether the plan has been executed sufficiently and whether the new enterprise architecture meets the expected deliverables. Within the act phase the adaption of targets and objectives based on the results from the check phase takes place. They influence the next plan phase by constraining direction and prioritization of future changes.

From an enterprise architect`s perspective, within the check phase it is important to understand which parts of the to-be were realized and which were not. Often, a KPI-based approach is suggested for the check phase. However, this presumes the existence of meaningful aggregations of observable properties into higher order EAM KPIs. The meaningful aggregation requires profound experience and a high level of EAM maturity, preconditions that we cannot always take for granted.

Therefore, we re-use gap analysis in the check phase as mechanism to derive closed, remaining and unconsidered gaps from existing enterprise architecture models. Gap analysis focusses on the EA models themselves, without aggregating observable properties into higher order properties.

In the plan phase, gap analysis compares two models. Here, we use it to compare three models during the check phase. The compared models are an old as-is, a new as-is and to-be as it was planned.

In our solution, the EA models are sets. Resulting subsets from the comparison are created through the set operators intersection, union and complement. The seven subsets form a disjoint decomposition of the EA models, partitioning the EA models into those subsets that were realized as planned, not realized as planned, realized but not planned and so on. They give the enterprise architect the necessary overview to conduct further analyses into the success of the plan execution.

2 Models as Sets

In our solution the core of an EA model is a set which consists of three different types of elements. The EA model contains the architecture building blocks of the EA, relations between architecture building blocks and attributes of architecture building blocks.

In this sense an EA model can be defined as $M := B \cup R \cup A$, where $B := \{x \mid x \text{ is an architecture building block}\}$, $R := \{x \mid x \in B \times B\}$ and $A := \{x \mid x\ B \times L\}$ and L is a set of possible attribute values. Additionally, the EA model comprises functions to structure the building blocks of the set, for example a naming function or partitioning functions, which allow for constructing hierarchies of types within the EA model. However, for the purpose of this paper we only need the core EA model, but not the additional functions.

Architecture building blocks stand for the elements of the enterprise architecture, for instance a CRM system in the application architecture. Relations hold between these architecture building blocks, for example when a system depends on another system the respective building blocks are connected by a dependency relation. Attributes are values associated with architecture building blocks that characterize measureable and observable characteristics of the architecture building block, e.g. the release number of an application component or the uptime of a service.

A distinction between observable and controllable variables is possible [1]. In this paper we focus on the observable variables derivable from the models already created. Whether, they are controllable is out of the scope of this paper.

2.1 Common Terminology for the different sets

In this subsection we provide a definition of the terms as used within this paper to avoid misinterpretations and create a common terminology [14].

- An *as-is model* is the set of the architectural building blocks, relations between them and their attributes as modeled and maintained at present. It serves as the planning basis within an enterprise architecture planning processes. At a later point in time it becomes the *Old-as-is* and is archived.
- A *to-be model* is the set of architectural building blocks, relations between them and their attributes as modeled for a future point in time. It can be used to express a desired target architecture which may have intermediate to-be`s as plateaus between itself and the as-is. A to-be model may become

the *New-as-is* if it is realized or may be adapted to a will-be as unplanned changes may occur and trigger an update of the to-be [3]. Furthermore, it is possible that a to-be is dismissed and another more preferable to-be is realized. As a consequence to-be models can be related to a certain point in time.

- *Gap analysis* (Delta Analysis) compares two EA models regarding differences. Comparisons considered in literature are: as-is and to-be, as-is and intermediate to-be, intermediate to-be and intermediate to-be, intermediate to-be and intermediate to-be [5].
- A *Gap* is a difference between two EA models. This is in accordance with TOGAF`s definition of a gap as “a statement of difference between two states” ([16], p. 357) and ArchiMate`s definition of a gap “as an outcome of a gap analysis between two plateaus” ([15], p. 148). After a gap has been identified it is possible to decide upon gaps to be removed (as-is and to-be, to-be (t_x) and to-be (t_{x+1})) or compare them (to-be alternative 1 (t_y) and to-be alternative 2 (t_y)). In contrast to ([16], chapter 27) we consider gaps not only as deleted or created building blocks, but also as changes in terms of removed and created relationships and attributes, as suggested in [7].

2.2 Defining the EA models as Sets

In this subsection we introduce the *Old-as-is*, the *As-was-planned* and *New-as-is* model that are analyzed using the gap analysis.

The EA metamodel restricts the modeling capabilities in models to certain types of architectural building blocks, relationships and attributes. Labusch and Winter [10] provide an overview of such a restriction to represent information relevant for enterprise architectures. The TOGAF content metamodel ([16], chapter 34) is a similar set of building blocks. Examples for building block types are Capability, Process, Business Object, (Logical and Physical) Application Component, Data Entity and (Logical and Physical) Technology Component.

We define the model *Old-as-is* as the as-is EA model that was used as the planning basis at a former point in time. The *Old-as-is* was analyzed and a desired future model was proposed and agreed upon from the stakeholders involved in the planning process. We call the desired future model the *As-was-planned* EA model. When a new planning cycle is initiated the *Old-as-is* model is updated and results in a model that we call the *New-as-is* EA model.

2.3 Defining the Different Sets created by the Gap Analysis

In the following we give mathematical definitions of the derivable subsets using three set operators to define each subset. These operators are *Union*, *Intersection* and *Complement* (Negation of set membership). With the *Old-as-is*, *As-was-planned* and *New-as-is* EA models at hand we define the following sets:

$$O1 = Old\text{-}As\text{-}Is \setminus (As\text{-}Was\text{-}Planned \cup New\text{-}As\text{-}Is) \tag{1}$$

O1 are building blocks, relationships and attributes which belong to the *Old-as-is* but are not in the *As-was-planned* and *New-as-is*. An example could be an application component which was retired as planned until the *New-as-is* was created.

$$S1 = (Old\text{-}As\text{-}Is \cap As\text{-}Was\text{-}Planned) \setminus New\text{-}As\text{-}Is \tag{2}$$

S1 are building blocks, relationships and attributes which the *Old-as-is* and the *As-was-planned* have in common, but not belong to the *New-as-is*. These were planned to be present in the furture, however they are not.

$$S2 = Old\text{-}As\text{-}Is \cap As\text{-}Was\text{-}Planned \cap New\text{-}As\text{-}Is \tag{3}$$

S2 are building blocks, relationships and attributes which were not planned to change and remained stable.

$$S3 = (Old\text{-}As\text{-}Is \cap New\text{-}As\text{-}Is) \setminus As\text{-}Was\text{-}Planned \tag{4}$$

S3 are building blocks, relationships and attributes which were planned to be changed, but were kept as they belong to the *Old-as-is* and the *New-as-is*.

$$N1 = As\text{-}Was\text{-}Planned \setminus (Old\text{-}As\text{-}Is \cup New\text{-}As\text{-}Is) \tag{5}$$

N1 are building blocks, relationships and attributes which were planned to be changed, but were not.

$$N2 = (As\text{-}Was\text{-}Planned \cap New\text{-}As\text{-}Is) \setminus Old\text{-}As\text{-}Is \tag{6}$$

N2 are building blocks, relationships and attributes which were changed as planned.

$$N3 = New\text{-}As\text{-}Is \setminus (Old\text{-}As\text{-}Is \cup As\text{-}Was\text{-}Planned) \tag{7}$$

N3 are building blocks, relationships and attributes which were realized but this was not foreseen in the plan.

Figure 1 shows an overview of the different sets.

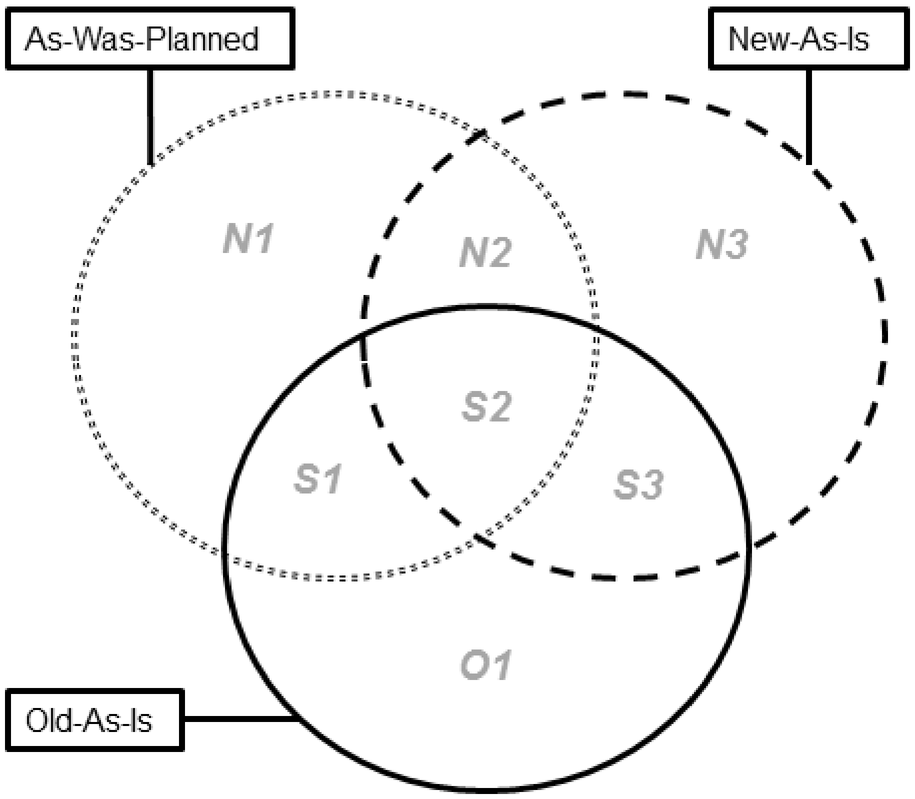

Figure 1: Different sets derivable from the sets Old-as-is, As-was-planned and New-as-is

3 Gap Analysis Applied to a Master Data Management Example

The sets presented in subsection 2.3 are now concretized with an example. The example is based on an excerpt of a research and development division`s application components, information system services, business services and business building blocks. The example provided in this paper uses development master data management (DMDM), which is a specialization of the master data challenge [11] for a research and development division. Its relevance from an EAM perspective is consolidating the development master data within one strategically defined application component and decreasing the functional redundancies present in the current architecture. Furthermore, the enterprise wants to integrate the development processes and reduce development costs, because manual double input of data and resulting inconsistencies are challenges that influence the strategic development of the application architecture. Technical constraints from a technology architecture are not considered within the example.

3.1 Architectural Building Blocks

In our example we have four types of architectural building blocks: (1) business building blocks, (2) business services, (3) application components and (4) information system services.

A *Business Building Block* (BBB) is an object which is created and changed by business activities of an enterprise regardless of process flows or domains [9]. Furthermore, a BBB has to be relevant for the value chain of the enterprise. It is implementation independent and there exists some authority which is responsible for it. Such an authority may be a project, an organizational unit or an application component.

A *Business Service*[1] supports the execution of business processes and uses BBBs. Furthermore, it can be implemented by information system services, partially or fully automated.

An *Application Component* is used by stakeholder groups within the enterprise to support their work in various ways. Furthermore, it is implemented on a technology stacks and need to be deployed on hardware to be ready for use.

An *Information System Service* is provided by an application component and can be consumed by other application components. Information system services are symbolized by the ‘Lollipop’-Symbol in Figure 2. Application components that use information system services are attached with them through the grappler symbol, indicating the coupling between them.

We consider for each model only building blocks that are actually present for that point in time. This is in accordance with Aier and Gleichauf [2]. If for example an *Application Component “Product planning tool”* changes its lifecycle from *live* to *retired,* we only include it in the architecture models where it has the attribute live. Our solution supports both ways of considering retired architectural building blocks in the models.

[1] Business Services are not part of the example as we consider only changes to the application components and their coupling via information system services. Nevertheless, we introduce them to avoid misunderstandings regarding the information system services.

3.2 Introducing the Old-as-is and As-was-planned

Figure 2 shows the application components, their implemented and used information systems services for the DMDM example. Furthermore, the BBB *Bill of Material* is shown whose responsible authority is the *Product class A assistance tool* in the Old-as-is. Further BBBs are not depicted in Figure 2.

The BBB *Bill of Material* is also associated with the *Product class B assistance tool* for another product configuration. This redundancy has historically grown and it was decided to unite both application components within a new *Product modification assistance tool*. As a consequence this new application component was planned and modeled within the as-was-planned model.

The information system services *MasterData_v1* and *MasterData_v2* were also planned to be consolidated within a single service *MasterData_v3*.

For the *Product planning tool* and the *Quality test planning tool* also a consolidation by introducing a new *Product and Quality test planning tool* was planned. This change included the development of an information system service *PlanningData_v1* which should be used by the *Quality test assistance and result management tool*.

Furthermore, the replacement of the *Virtual quality test result tool* by *Quality test assistance and result management tool* was planned including the dependency to the information system service *PlanningData_v1*.

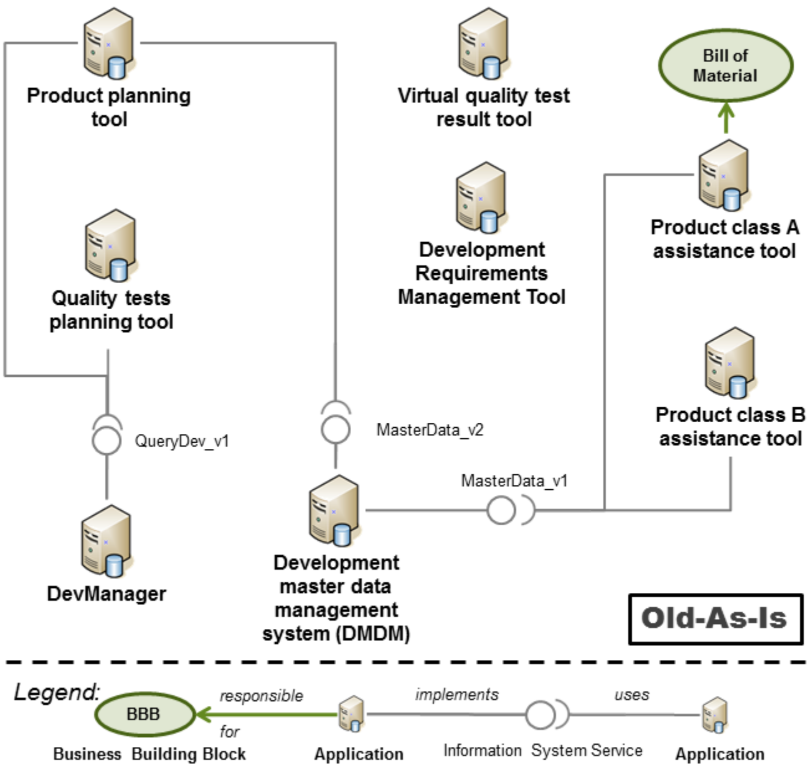
Product planning tool
Virtual quality test result tool
Bill of Material
Product class A assistance tool
Quality tests planning tool
Development Requirements Management Tool
Product class B assistance tool
QueryDev_v1
MasterData_v2
MasterData_v1
DevManager
Development master data management system (DMDM)
Old-As-Is
Legend:
BBB
responsible for
implements
uses
Business Building Block
Application
Information System Service
Application

Figure 2: Visualization of the Old-as-is

No phasing out was planned for the *Development master data management system*, the *DevManager* its implemented information system service *QueryDev_v1* and the *Development Requirements Management Tool*.

3.3 Updating the as-is – Creating the New-as-is

Within the next planning phase the as-is is updated and the *New-as-is* is modeled. The BBBs have remained stable.

Product and Quality test planning tool, *PlanningData_v1*, *Quality test assistance and result management tool*, *MasterData_v3* were realized as planned and thus were modeled within the new-as-is. The *Development master data management system* is still present, as planned, in the new-as-is. However, the information system service *MasterData_v1* is still present even though it was planned to be retired. This extends to the application components *Product class A assistance tool* and *Product class B assistance tool*, which still use the information system service *MasterData_v1*.

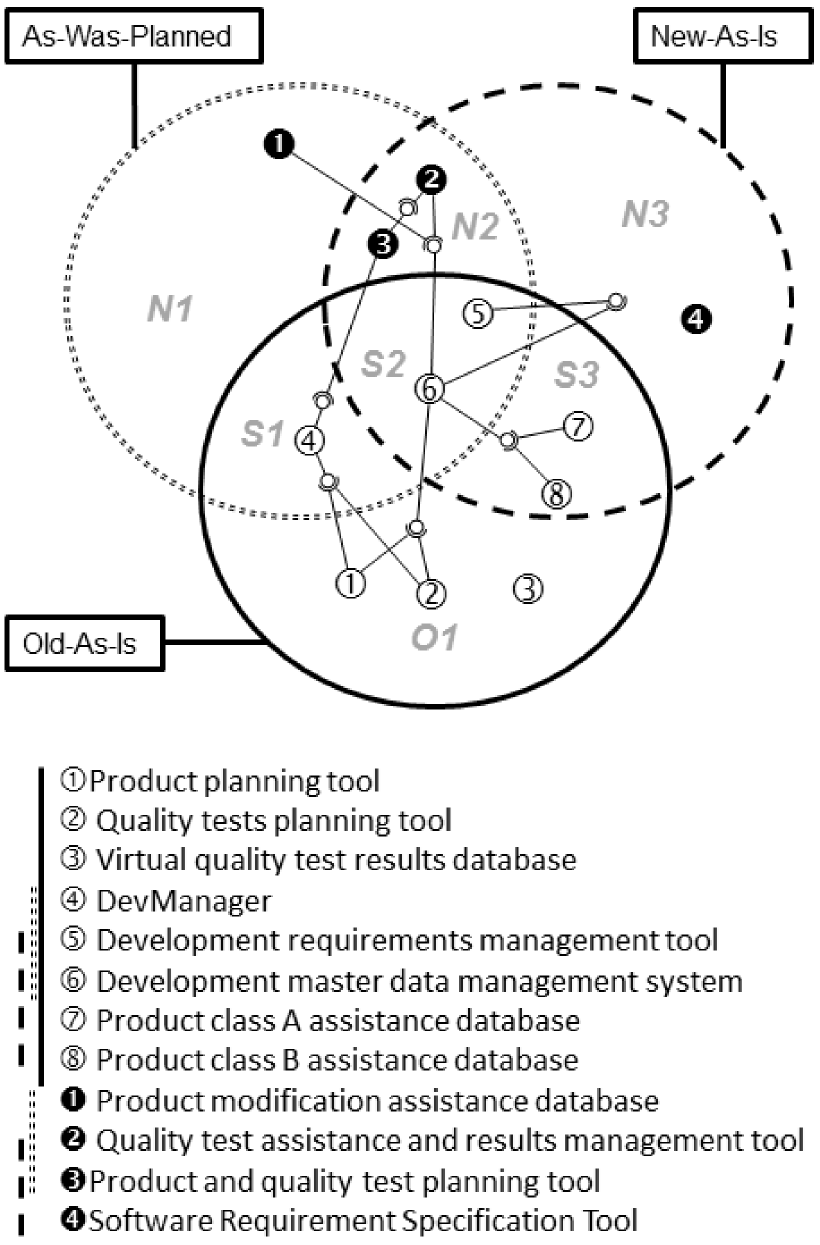
As-Was-Planned
New-As-Is
Old-As-Is
N1
N2
N3
S1
S2
S3
O1
❶
❷
❸
❹
①
②
③
④
⑤
⑥
⑦
⑧
① Product planning tool
② Quality tests planning tool
③ Virtual quality test results database
④ DevManager
⑤ Development requirements management tool
⑥ Development master data management system
⑦ Product class A assistance database
⑧ Product class B assistance database
❶ Product modification assistance database
❷ Quality test assistance and results management tool
❸ Product and quality test planning tool
❹ Software Requirement Specification Tool

Figure 3: Resulting Sets for the DMDM Example

There are two new architectural building blocks in the new-as-is model: an information system service *SpecificationData_v1* implemented by the *Development Requirements Management Tool* and a new application component *Software Requirement Specification Tool*.

Furthermore, the application component *DevManager* including the implemented information system service *QueryDev_v1* were retired without being planned to be retired.

Figure 3 shows an overview of the resulting sets for the DMDM example.

This information can be provided to the enterprise architect. The stable building blocks are at the core of this illustration. The more volatile building blocks (those retired, those new, and those that should have been retired but were not) are in the peripheral areas (O1, N1, and N3) in the illustration. As a further orientation, architectural building blocks that were in the *Old-as-is* are white, the others are black in Figure 3.

3.4 Further Analysis by the Enterprise Architect

The enterprise architect can now easily identify the crucial components, for example those that were realized but not planned in the set N3. In the DMDM example there are two building blocks in N3: the information system service *SpecificationData_v1* and the *Software Requirement Specification Tool*. The enterprise architect decides which gaps are crucial and which are not. As a consequence the meaning of each gap does vary between different EAM approaches. However, it is possible to assist in finding such crucial gaps, if information regarding the types of elements and relations between those is available, by providing filter options for the gaps.

The implementation of a new information system service was, from an enterprise architects perspective, deemed not problematic. There is no (application) architecture principle present restricting the development of information system services, as long as they do not collide with existing responsibilities for business building blocks and consider technical constraints imposed by the EAM. The development of the new *Software Requirement Specification Tool* was seen as problematic and was subsequently investigated in more detail by the enterprise architect.

4 Discussion

In this section we discuss the results and limitations of using the gap analysis to support the check phase of enterprise architecture management.

4.1 Results

All Gaps which were closed are within the subset N2. Furthermore, we were able to identify a relation between subsets of N1 and subsets of S3 namely the unrealized changes. In our example there is only one subset for N1 and S3 each. However, it is possible that several subsets exist within N1 and S3 which may interrelate with each other.

Architectural building blocks, relationships and attributes from N3 do not necessarily have a relationship to those in S1. The reason for that is that N3 can contain unplanned and problematic changes from an EAM perspective. However, it can also contain unplanned but unproblematic changes.

A benefit from the proposed use of the gap analysis is that it works for different timeframes in the same way, as long as the metamodel remains stable. As a consequence, the use of the gap analysis needs to be extended for situations with architectural revolutions (architecture paradigm changes).

Even though we presented an example case with predefined architectural building blocks, relationships and attributes the solution is EA metamodel independent. Any stakeholder who needs information on closed, not closed and unintentionally closed gaps can gather it through the proposed solution.

4.2 Limitations

The documentation of enterprise architecture models, regardless if (semi-)automatically or manually generated by different stakeholders, is one of the core processes of enterprise architecture management [13]. However, not every EAM approach creates to-be models (c. f. [17]) and as a consequence the proposed usage of the gap analysis is only possible if a to-be model is available.

Furthermore, we assume the same level of detail for as-is and to-be models regarding the architectural building blocks, relationships and attributes. This shortcoming could be leveraged by using advanced analysis strategies like suggested by Binz et al. [4] for enterprise topologies to create abstracted EA models. However, this assumption seems to be a minor shortcoming as the comparability of to-be models is a prerequisite for supporting planning decisions ([2], [3] and [5]) and a to-be model can become the new as is.

Finally, the set theoretic implementation of gap analysis relies on sets as inputs. Consequently, the EA models have to be either sets, or the relevant aspects of the EA model have to be projected into sets. The latter is the case if the EA model is, for instance, a logical theory. Then, individuals and relations between them become the elements of the set used for gap analysis. Type assertions and further logical constraints that are relevant for modeling an EA can be excluded if they are deemed unnecessary for the gap analysis.

5 Related Work

Murer et al. [12] present an EAM approach which tries to strike a balance between business and information technology developments which they call managed evolution. They make a distinction between properties which are important for ever and properties "related to gaps between reality and target architecture" ([12], p.211). Latter are the ones which allow an enterprise architect to figure out which changes filled gaps and which gaps were not closed.

Hanschke [8] uses the gap analysis to detect differences between a current and target architecture to derive planned information system architecture (intermediate to-be). The gaps are detected by comparing the business support maps of the current and target architecture. Business support maps localize information systems by relating them to business processes and organization units. The detected gaps and their interdependencies are analyzed, ranked and clustered together in bundles of gaps. The decision to resolve one or more bundles results in an intermediate to-be model, which will be realized by projects. Gaps of other domains than the information system architecture are not considered.

TOGAF ([16], chapter 27) provides an introduction to the gap analysis as key ingredient at the end of the phases B, C and D of its *Architecture Development Method* (ADM). As a consequence gaps can relate to the business, data, application and technology domain. As a procedure to identify gaps

TOGAF suggests drawing up a matrix with all building blocks of the to-be model on the horizontal axis and all building blocks of the as-is on the vertical axis. This approach has the shortcoming that it only considers added and removed building blocks but not how to cope with changed building blocks, even though impacted building blocks are considered as important. Furthermore, *data relationship gaps* are considered as an important outcome of the gap analysis, but a description how to derive these gaps is missing.

ArchiMate ([15], chapter 11) introduces an *Implementation and Migration Extension* including a *Gap* element. A gap can be *associated with* any core element of the ArchiMate metamodels, except for the *Value* and *Meaning* element. In general, a gap links several elements of two EA models and contains elements to be removed (retired) and to be added (developed). The linkage of the differences between the EA models and the resulting gap is not described.

6 Future Work

We are not considering new gaps within this paper which arise due to changes of to-be models which are still to realize. These could be derived by performing the gap analysis between the new-as-is model and one or more new-to-be models. Future work should address this comparison as well as the resulting subsets.

Furthermore, from an enterprise modeling perspective the identification of dependencies between closed gaps and open gaps should be supported. With the subsets at hand it is possible to start searching for subsubsets where the different as-is and to-be models diverge and how these different subsubsets relate to each other. The presented subsets serve as the starting point for further investigations through an enterprise architect.

We plan to realize the gap analysis as proposed in this paper in a prototypical tool which will provide the described subsets and further query mechanisms to an enterprise architect. This technical realization will include the steps

- Create *Old-as-is* (Enterprise architect; for t0 at t0)
- Copy *Old-as-is* (Tool)
- Model *to-be* based on Copy *Old-as-is* (Enterprise architect; for t1 at t0)
- Copy *Old-as-is* (Tool)
- Create *New-as-is* based on Copy of *Old-as-is*(Enterprise architect; for t1 at t1)
- Perform Gap Analysis and show subsets (Tool)
- Examine subsets (Enterprise architect)
- Draw conclusions for modeling New-to-be (Enterprise architect)

Furthermore, we want to check the applicability of the solution to support an enterprise architect in the creation of a *will-be model* [3].

If information about the successor relationships between architectural building blocks is available it could be used for root cause analyses for gap deviations. However, capturing these relationships between architectural building blocks for different points requires a transformation model [2].

7 Conclusion

The paper presents a solution that uses gap analysis within the check phase of EAM by comparing three different EA models. Based on this comparison it is possible to identify gaps which were closed and which remain open with respect to a plan.

The sets which make up the different models were defined and their resulting subsets were presented. Afterwards, the subsets were applied on an example of the master data management challenge for a research and development division.

8 Literature

[1] Abraham, R; Tribolet, J; Winter, R (2013): Transformation of Multi-level Systems – Theoretical Grounding and Consequences for Enterprise Architecture Management. In: Proper, Henderik A.; Aveiro, David; Gaaloul, Khaled (Eds.) - *Advances in Enterprise Engineering VII*. Springer, Berlin.

[2] Aier, S; Gleichauf, B (2010): Application of Enterprise Models for Engineering Enterprise Transformation. In: Enterprise Modelling and Information Systems Architectures 5(1): 56–72.

[3] Aier, S; Gleichauf, B; Saat, J; Winter, R (2009): Complexity Levels of Representing Dynamics in EA Planning. In: Albani, Antonia; Barjis, Joseph; Dietz, Jan L. G. (Eds.) – *Advances in Enterprise Engineering III*. Springer, Berlin.

[4] Binz, T; Leymann, F; Nowak, A; Schumm, D (2012): Improving the Manageability of Enterprise Topologies Through Segmentation, Graph Transformation, and Analysis Strategies. In: Proceedings of 16th IEEE International Enterprise Distributed Object Computing Conference (EDOC 2012): p. 61–70.

[5] Buckl, S; Schweda, CM (2011): On the State-of-the-Art in Enterprise Architecture Management Literature. Technical Report, Technische Universität München.

[6] Deming, WE (1994): Out of the crisis – quality, productivity and competitive position. 19th Edition. Cambridge University Press, Cambridge.

[7] Diefenthaler, P; Bauer, B (2013): Gap Analysis in Enterprise Architecture Using Semantic Web Technologies. In: Proceedings of the 15th International Conference on Enterprise Information Systems (ICEIS 2013). Angers, France.

[8] Hanschke, I (2013): Strategisches Management der IT-Landschaft – Ein Praktischer Leitfaden für das Enterprise Architecture Management. 3. Edition. Carl Hanser Verlag, München.

[9] Hess, C; Lautenbacher, F; Fehlner, K (2013): Business Building Blocks as Coordination Mechanism for Enterprise Transformations. 17th IEEE International Enterprise Distributed Object Computing Conference Workshops (EDOCW), Vancouver, Canada, in print.

[10] Labusch, N; Winter, R (2013): Towards a Conceptualization of Architectural Support for Enterprise Transformation. In: Proceedings of the European Conference on Information Systems (ECIS) 2013. - ECIS 2013. - Utrecht, Netherlands, Paper 137.

[11] Loshin, D (2009): Master Data Management. Morgan Kaufmann, Amsterdam.

[12] Murer, S; Bonati, B; Furrer, F (2011): Managed Evolution – A Strategy for Very Large Information Systems. Springer, Berlin.

[13] Op’t Land, M; Proper, HA; Waage, M; Cloo, J; Steghuis, C (2009): Enterprise Architecture - Creating Value by Informed Governance. Springer, Berlin.

[14] Schönherr, Marten (2009): Towards a Common Terminology in the Discipline of Enterprise Architecture. In: Feuerlicht, George; Lamersdorf, Winfried (Eds.) - Service-oriented computing, ICSOC 2008 workshops. Springer, Berlin.

[15] The Open Group (2012): ArchiMate 2.0 Specification. The Open Group Series, Van Haren Publishing.

[16] The Open Group (2011): TOGAF Version 9.1. The TOGAF Series, Van Haren Publishing.

[17] Winter, K; Buckl, S; Matthes, F; Schweda, CM (2010): Investigating the state-of-the-art in enterprise architecture management method in literature and practice. In: 5th Mediterranean Conference on Information Systems (MCIS2010), Tel Aviv, paper 90.