Patterns of Evolution in Enterprise Architecture Information Models

Sabine Buckl, Christian M. Schweda
iteratec GmbH, {Sabine.Buckl—Christian.Schweda}@iteratec.de

Abstract: *Enterprise architecture (EA) models* are key artifacts for managing the development of the enterprise and in particular its IT landscape, i.e. are central to EA management. The EA models are used to document the current state of the enterprise according to the information demands of EA stakeholders. These demands are reflected in the architecture models' corresponding meta-models, the *EA information models*. With rising maturity of EA management the group of stakeholders as well as the stakeholders' information demands change. Further, EA management is embedded into a changing environment, where recent regulatory constraints on the one hand and novel technological paradigms keep changing the perspective on the enterprise. Consequently, the EA information models are adapted to accommodate the changed requirements.

In this paper, we revisit existing approaches in the area of EA information modeling and discuss the identified lack of support regarding information model evolution. To close the experienced gap, we discuss recurring patterns of adaptation that we observed in the field of EA modeling.

1 Motivation

Enterprise Architecture (EA) management takes an overarching and embracing perspective on the enterprise to identify and leverage advantages that would otherwise be lost in the silos of isolated management activities or organizational islands. EA management for example brings together information from business process management with information from IT infrastructure management to facilitate discussions on the impact of hardware outages on the business continuity. Models of the EA are used to support the management activities and cover manifold different aspects of the enterprise. In particular, the models support documentation, communication, and analysis of the current state of the EA as well as the development of planned states. The corresponding meta-models, called *EA information models*, define the concepts relevant for modeling the EA and supply a consistent terminology ranging from business concepts to infrastructure elements.

EA management is enterprise-specific in particular regarding the coverage of the enterprise. The organizational set-up and the empowerment of the EA management activity greatly influences which aspects of the enterprise can be considered from a holistic perspective, and which are (currently) out-of-scope. Both covering the concepts, which are in the outreach of the current EA management, in the EA information model and constraining the information model to only these concepts are two critical success factors. On the one hand

failing to provide information about a part of the EA relevant to a key stakeholder leads to quickly decreasing sponsorship by this stakeholder. On the other hand an attempt to anticipate concepts, which 'may be of use some day', leads to an "oversized information model" [BEMS09] that increases the expenses for information retrieval and data maintenance and thus can put the whole EA management endeavor at risk. Rising maturity of EA management may require to involve additional stakeholders, as may changing legal regulations and technological paradigms affect the concerns of existing stakeholders. The advent of the paradigm of service-orientation, for example, raised additional demands for EA modeling as reflected in the survey of Matthes et al. [MBLS08].

Current frameworks for EA management do not specifically account for this need to co-evolve their object-oriented information models with the evolving and maturing EA management function, as our analysis of the state-of-the-art in Section 2 indicates. This evolution nevertheless is not to be mistaken for a set of arbitrary information model adaptations, which are simultaneously applied to the EA models. Using Guizzardi's foundational ontology [Gui05] (see Section 3) to abstract from representational peculiarities in EA information models, we identify distinct *evolution patterns* for EA information models. These patterns, presented in Section 4, are on the one hand means to simplify the information model evolution, which should be supported by EA management tools. On the other hand, the patterns provide an increased level of abstraction for describing recurring evolution steps together with the typical rationalizations, i.e. supporting arguments and objections. Final Section 5 outlines directions for future research building on the identified patterns and establishes a promising link to the field of EA model visualization.

2 EA information model evolution in current EA management frameworks

In an extensive literature review [BS11], Buckl and Schweda analyze 22 EA management frameworks regarding both their provided best-practices for EA management processes and for EA information models. One particularly interesting dimension is the one of "configure and adapt" for the described information models. Buckl and Schweda distinguish between approaches that

- provide *no mechanisms* for tailoring the EA information model to the concerns of the using organization,
- delineate, or at least outline, mechanisms for *initially configuring* an EA information model for the using organization, and
- describe, or at least sketch, mechanisms for *adapting* an EA information model to changing organizational concerns.

Their analysis reveals that, while most of the approaches acknowledge the need for enterprise-specific EA modeling, only twelve (55%) of the frameworks and approaches discuss mechanisms for configuring and/or adapting the EA information model. Moreover, only three

of these twelve approaches (14% of the analyzed frameworks) discuss mechanisms for adapting the EA information model after the initial configuration. In the following, we briefly revisit these three frameworks together with The Open Group Architecture Framework TOGAF [The09] and the Best-Practice EA of Hanschke [Han10]. Former framework can be considered a 'de-facto standard' in the field of EA management, whereas latter framework is exemplary for tool-supported frameworks.

The *extended enterprise architecture* framework has been developed by the Institute for Enterprise Architecture Developments since 2002. The framework discusses that stakeholders' goals shape the structure of an EA information model. In [Sch08] Schekkerman sketches that each goal is complemented by a particular part of an information model and that with changing goals, selected parts of the information model can be added and removed. Nevertheless, the framework does not discuss the implications of adding and removing information model fragments, nor are changes in the perspective on architecture aspects discussed.

The *semantic object model* developed by Ferstl and Sinz [FS95] introduces an information model centering around the concepts *business object* and *business transaction*. Based on these concepts the business perspective on an organization's EA is described. Acknowledging the fact that with changing stakeholders' needs also the required level of abstraction changes, the semantic object model introduces rules of *decomposition*. One such rule, for example, describes how a business object can be decomposed into two business objects that are optionally linked by a business transaction. The provided rules are confined to the object-level and do not entail mechanisms to introduce additional concepts to the information model.

Kurpjuweit and Winter discuss in [KW09] the need to create organization-specific information models. Concern-specific model fragments form the basis for the creation of such models by integration. In [Kur09] Kurpjuweit makes available a collection of such fragments together with concern-specific viewpoints onto these. Possible evolutions of an information model can be achieved by exchanging different fragments that address related concerns. Nevertheless, no discussions on consequences of switching between concerns or viewpoints are undertaken.

The Open Group Architecture Framework [The09] supplies the *enterprise content metamodel*, which is an information model for EA management. This model decomposes into a core model supplying key concepts relevant for most organizations and several *extensions*, e.g. for modeling services or motivational aspects like goals. These extensions do not change the structure of the core information model but provide additional concepts that are linked via optional relationships to key concepts. Textual descriptions along the service extension, for example, discuss consistency issues that may arise from applying both the complete core model and the service extension, instead of confining 'usage modeling' between business processes and information systems to either one way of modeling. TOGAF does further not describe how existing reports are affected, if an extension is applied.

The Best-Practice EA of Hanschke [Han10] introduces an information model for EA management, which is intended to be adapted to the specific concerns of the using organization's stakeholders. As means of adaptation, the omission of concepts as well as adding

and removing properties are discussed. Further, Hanschke describes that the predefined relationships can be refined by adding enumeration-typed properties that detail the semantics of an actual instantiation. In this context, the topic of possible inconsistencies is not discussed in much detail. For all aforementioned types of information model adaptations, Hanschke briefly describes advantages and disadvantages. Further, the best-practice EA states that several of the contained concepts are *hierarchic*, which means that corresponding instances can be decomposed. Thereby, an instance-level refinement mechanism is provided.

In summary, we agree with Buckl and Schweda, who state in [BS11], that the need for adaptation is acknowledged by current frameworks, while actual mechanisms to perform this adaptations are scarce. Additionally, we diagnose that even for existing adaptation mechanisms the documentation of the necessary context for the application and the consequences of applying the adaptation is non-existent.

3 Related work: foundational ontology for conceptual modeling

In his work [Gui05] Guizzardi establishes ontological foundations for conceptual modeling. Building on the basic type-instance dichotomy of object-oriented modeling languages, he proposes several clarifications and refinements that contribute to an ontologically sound perspective for modeling. Central to this perspective is a distinction of different kinds of types. Guizzardi does not simply identify type, which he calls *universal*, with "class", but distinguishes between:

Substantial universals whose corresponding instances are individuals which have an existence in their own.

Relationship universals which are instantiated to links that exist as reifications of relations between individuals.

Regarding the substantial universals, Guizzardi continues the discussions of Guarino and Welti [GW00] on the meta-property of *rigidity* for types. A type is considered rigid, if being of this type is an *essential* property of all corresponding instance. This means, that any individual $i \in \mathcal{I}$ being instance of a type T once means being instance of that particular type ever, formally:

$$\forall i \in \mathcal{I} : \; i :: T \Rightarrow \Box \, i :: T.$$

Non-rigid conversely means that at least one individual $i \in \mathcal{I}$ is instance of a type T at one point in time but not forever, formally:

$$\exists i \in \mathcal{I} : \; i :: T \wedge \neg \Box \, i :: T.$$

For our considerations, a particular kind of non-rigid substantial universals is of interest. Guizzardi calls them *Phase*s, which are *anti-rigid*. Anti-rigid means that any individual which is once an instance of a given type, stops to be an instance of that type at at least

one point in time, formally:

$$\forall i \in \mathcal{I} : \; i :: T \Rightarrow \neg\Box \; i :: T.$$

Phases, i.e. anti-rigid types, can be used to model that with the change of the characterization of an individual, additional characterizations (via properties and relationships) become available and/or are not longer valid.

4 Patterns of EA information model evolution

The subsequently described evolutions of EA information models have been observed in practice, where they are usually applied to address a particular modeling *problem*. The evolution itself describes a *solution* to the given problem, leading to side-effects that form the *consequences* of the application aside the information model being adapted. Acknowledging the nature of documented evolutions as *repeatedly observed practice-proven solutions to relevant problems*, we document the typical evolutions as *evolution patterns* with the typical structure of a pattern (cf. Meszaros and Doble [MD97]).

4.1 Life-cycle Documentation

After an initial documentation phase in which mostly the as-is state of the enterprise is documented, a maturing EA management seeks to get an overview about already planned changes. A first step in planning involves a refined documentation of the life-cycle state in which particular architecture elements are.

Context An enterprise has already established a (mandatory) documentation of the life-cycle state of architecture elements in the as-is state. Typically, life-cycle modeling applies to elements like *information systems*, *technical components* and *hardware devices*, i.e. deployable or physically deployed elements.

Problem Architecture elements, whose development over time should be documented, undergo different life-cycle changes. A more detailed planning of the life-cycle is necessary to express, that elements in different states have additional characteristics, e.g. users or roles responsible during the particular phase in the life-cycle. The basic modeling of life-cycle as a (mandatory) property in the given substantial type fails to cover these dependencies to additional characteristics.

Solution For the affected type of architecture elements (cf. Figure 1), anti-rigid sub-types are introduced with each type reflecting one particular life-cycle state. The (mandatory) enumeration property that was used to indicate the life-cycle state becomes a derived property, whose values are fixed in the anti-rigid sub-types. The original type of architecture

elements is converted to an abstract type. Figure 2 describes the resulting information model fragment.

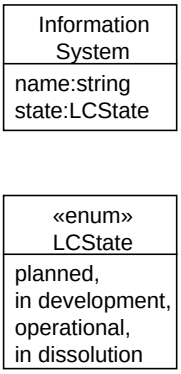

Abbildung 1: *Life-cycle documentation*: initial fragment

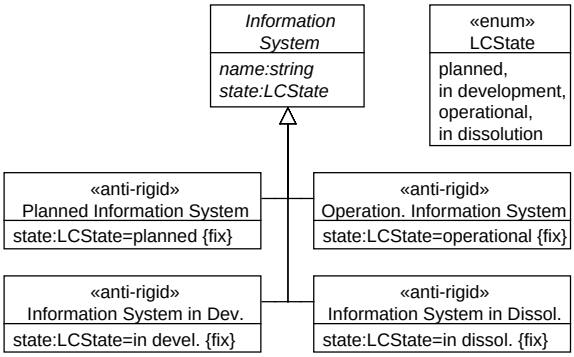

Abbildung 2: *Life-cycle documentation*: evolved fragment

Consequences The introduction of the anti-rigid sub-types allows to concisely model the characteristics of the architecture element in the different states of the life-cycle. Current analyses, reports, and visualizations building on the changed type and/or on the state of the life-cycle property are not affected. A potential exception would occur, if the life-cycle property was not mandatory. In such cases, corresponding instances that do not supply a value for the property can be retained as instances of the (then not abstract) original class with the property value fixed to `null`. This variation leaves analyses, reports, and visualization unaffected, but fails to provide an ontologically clear modeling.

4.2 Servicialization

Re-use of existing architecture elements in different contexts is a key factor of many recently emerging paradigms of enterprise planning, e.g. service-oriented architecture (SOA), virtualization, and cloud computing. Each of these paradigms entails a separation of 'function' independent from 'structure'.

Context An enterprise documents *use* or *supported by*-relationships between architecture elements in the EA. The number of *used* or *supporting* elements, e.g. *information systems* is considered large or hints, that the support, e.g. for *business processes*, provided by these elements was redundant, exist.

Problem Architecture elements which are used by or support other architecture elements, usually residing on a different architectural layer, are hard to compare based on the provided characteristics. The characteristics, e.g. properties, cover mostly non-functional and structural aspects of these architecture elements, for example component structure and performance properties for *information systems*. The supported or using elements, e.g. *business processes*, are too coarse grained to facilitate the necessary comparisons.

Solution Refine the *use* or *supported by*-relationship (cf. Figure 3) with an intermediary relationship type that describes the provided or used functionality. An additional type `Service` reifying the functionality is associated with the intermediary type. The intermediary type designates non-functional requirements of service utilization. For the initial *use* or *supported by*-relationship a derived relationship computing the corresponding information is provided. Figure 4 shows the evolved fragment.

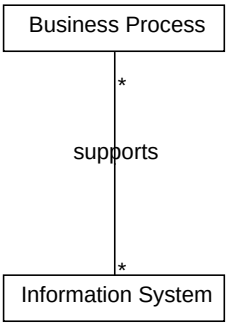

Abbildung 3: *Servicialization*: initial fragment

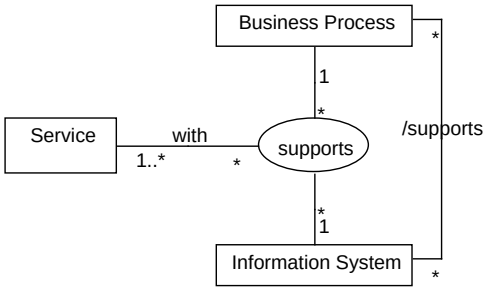

Abbildung 4: *Servicialization*: evolved fragment

Consequences The reification of the *use* or *supported by*-relationship to a relationship type does not affect existing analyses, reports, and visualizations, as the derived relationship can be used to supply the relevant information. The introduction of the type describing services does not come without costs, but leads to arguments regarding the appropriate level of granularity. Further, it can be considered a challenge to formulate the different services in a uniform terminology. Especially after the service-providing architecture elements have been in use for a longer period of time, it happens, that the name of the element has become a *common noun* for the provided service. In such cases, it is nevertheless advisable to introduce a differing name. The introduction of properties designating non-functional characteristics of the service utilization adds to modeling complexity, although it is considered crucial in eliciting which supporting architecture element should be used to provide a particular support.

4.3 Selective federation

In multiple recent business models as well as IT operations models, the enterprise does not longer provide and operate the relevant infrastructures as a whole. Other enterprises, like *outsourcing* providers or *outtasking* partners contribute parts of the infrastructure on different levels of the EA.

Context An enterprise established a network of outsourcing and outtasking organizations, providing critical infrastructure, like *information systems* or *supportive processes*, for the core business processes. The partnering organization is not owned or completely controlled by the enterprise, but is willing to participate in a federated EA management initiative.

Problem It is usually no binary decision whether to outsource/outtask or not. Contrariwise, infrastructure federation is usually performed selectively with certain parts of the EA being admissible for federation, whereas other parts – usually related to *unique features* of the enterprise – are kept in-house. The EA model must reflect this selective federation, making it possible to model in-house parts more in-depth, whereas other parts of the EA model, like external IS services, remain described in less detail.

Solution Several levels of *use* or *supported by*-relationships form the basic supportive infrastructure layering in an enterprise, see Figure 5. The layer on which selective federation takes place, is refined via a distinction between in-house elements and external elements. Existing self-relationships concerning the element type, which is subject to selective federation, are reassigned such that only relationships originating from an in-house element are described. Thereby, the enterprise retains ownership of all relevant intra-layer relationships that may impact the provided support. The base class on the particular layer is converted to an abstract class to ensure a sound modeling. In case selective federation is applied on different layers of the supportive infrastructure, the evolution as displayed in Figure 6 can be repeatedly used on each of these layers.

Consequences The distinction between in-house and external architecture elements on a particular layer allows to concisely describe, where federation takes place. Nevertheless, the evolution affects multiple analyses, reports, and visualizations, in particular ones that do not only include the given layer, but also the "lower" layer that is used by or supports the affected one. Furthermore, the evolved EA information model must be designed to facilitate actual federation of architectural knowledge with the partnering organization. This pertains in particular to the *identity conditions* for external elements. The partnering organization is most likely to use a different identification schema for the provided infrastructure, such that the external elements must bear multiple identifying properties both for internal and external referencing.

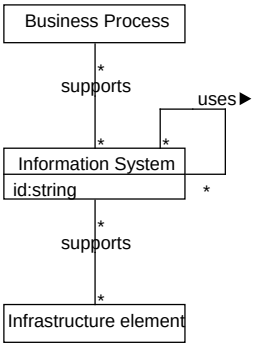

Abbildung 5: *Selective federation*: initial fragment

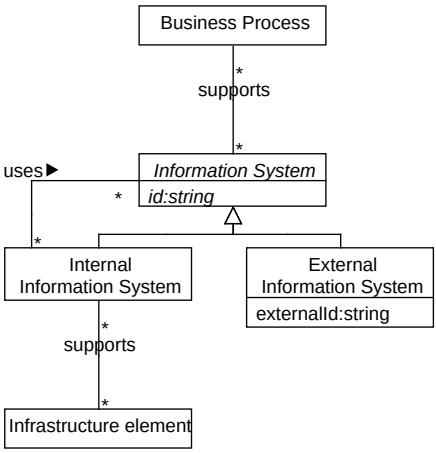

Abbildung 6: *Selective federation*: evolved fragment

5 Summary and Outlook

In this paper we discuss the topic of EA information model evolution. The EA information models are tailored to the information demands of the stakeholders involved in EA management. With changing stakeholders as well as changing concerns of EA management, also the information models have to evolve. We describe typical patterns of information model evolution as model transformations together with supporting arguments and objections against the particular kind of evolution.

The presented list of patterns is not comprehensive but exemplary and particularly aims at describing an initial set of evolutions. This set can be useful in re-design projects seeking to adapt the EA information model in response to changing concerns and stakeholders. In addition, the *patterns of evolution* may also be of interest in the context of EA visualizations. EA visualizations are used to convey information about the EA to relevant stakeholders. Different stakeholders use different kinds of visualizations, i.e. *viewpoints*, to perform particular tasks, like planning and analysis. Usually these visualizations do not convey all available information from the EA model, but apply abstractions. While some of these abstractions are merely filters and projections to particular types and properties, other abstractions can be formulated in model transformations [BEL+07]. As of today no catalog of typical "transformatory abstractions" exist. Some observed abstractions nevertheless resemble evolution patterns as documented in Section 4 or inverse applications of these patterns. This resemblance can be an interesting starting point for future research.

Literatur

[BEL+07] Sabine Buckl, Alexander M. Ernst, Josef Lankes, Kathrin Schneider und Christian M. Schweda. A pattern based Approach for constructing Enterprise Architecture Management Information Models. In A. Oberweis, C. Weinhardt, H. Gimpel, A. Koschmider, V. Pankratius und Schnizler, Hrsg., *Wirtschaftsinformatik 2007*, Seiten 145–162, Karlsruhe, Germany, 2007. Universitätsverlag Karlsruhe.

[BEMS09] Sabine Buckl, Alexander M. Ernst, Florian Matthes und Christian M. Schweda. How to make your enterprise architecture management endeavor fail! In *Pattern Languages of Programs 2009 (PLoP 2009), Chicago*, 2009.

[BS11] Sabine Buckl und Christian M. Schweda. On the state-of-the-art in EA management literature. Bericht, Chair for Informatics 19 (sebis), Technische Universität München, Munich, Germany, 2011.

[FS95] Otto K. Ferstl und Elmar J. Sinz. Der Ansatz des Semantischen Objektmodells (SOM) zur Modellierung von Geschäftsprozessen. *Wirtschaftsinformatik*, 37(3):209–220, 1995.

[Gui05] Giancarlo Guizzardi. *Ontological foundations for structural conceptual models*. Dissertation, CTIT, Centre for Telematics and Information Technology, Enschede, The Netherlands, 2005.

[GW00] Nicola Guarino und Christopher A. Welty. Identity, Unity, and Individuality: Towards a Formal Toolkit for Ontological Analysis. In Werner Horn, Hrsg., *Proceedings of the 14th European Conference on Artificial Intelligence*, Seiten 219–223, Berlin, Germany, 2000. IOS Press.

[Han10] Inge Hanschke. *Strategic IT Management – A Toolkit for Enterprise Architecture Management*. Springer, Berlin, Germany, 2010.

[Kur09] Stephan Kurpjuweit. *Stakeholder-orientierte Modellierung und Analyse der Unternehmensarchitektur*. Dissertation, Universität St.Gallen, 2009.

[KW09] Stephan Kurpjuweit und Robert Winter. Concern-oriented business architecture engineering. In *SAC'09: Proceedings of the 2009 ACM symposium on Applied Computing*, Seiten 265–272, New York, NY, USA, 2009. ACM.

[MBLS08] Florian Matthes, Sabine Buckl, Jana Leitel und Christian M. Schweda. *Enterprise Architecture Management Tool Survey 2008*. Chair for Informatics 19 (sebis), Technische Universität München, Munich, Germany, 2008.

[MD97] Gerard Meszaros und Jim Doble. *A Pattern Language for Pattern Writing*, Seiten 529–574. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1997.

[Sch08] Jaap Schekkerman. *Enterprise Architecture Good Practices Guide – How to Manage the Enterprise Architecture Practice*. Trafford Publishing, Victoria, BC, Canada, 2008.

[The09] The Open Group. TOGAF "Enterprise Edition" Version 9. `http://www.togaf.org` (cited 2011-06-08), 2009.