Implementing a Software Prototype for Enterprise Architecture Rationalization: Lessons Learned

Georgios Plataniotis  
Public Research Centre Henri Tudor, Luxembourg, Luxembourg  
Radboud University Nijmegen, Nijmegen, the Netherlands  
EE-Team, Luxembourg, Luxembourg  
georgios.plataniotis@tudor.lu

Sybren de Kinderen  
University of Luxembourg, Luxembourg  
EE-Team, Luxembourg, Luxembourg  
sybren.dekinderen@uni.lu

Henderik A. Proper  
Public Research Centre Henri Tudor, Luxembourg, Luxembourg  
Radboud University Nijmegen, Nijmegen, the Netherlands  
EE-Team, Luxembourg, Luxembourg  
erik.proper@tudor.lu

Abstract—Enterprise Architecture (EA) modeling languages describe an enterprise architecture holistically. Therefore, they show an enterprises business products and services, and how these are realized by IT infrastructure and applications. However, EA modeling languages lack the capability of capturing rationalization information behind these models in terms of selection criteria, alternatives etc.

Our earlier work proposes the EA Anamnesis approach for enterprise architecture rationalization. Its major contribution is a formal metamodel and a corresponding concrete syntax to interrelate business and IT decisions and in turn complement EA models with design rationale information. Yet, up to now the EA Anamnesis approach lacks software tool support.

In this paper we discuss the lessons learned during the implementation of our metamodel into a software prototype. Furthermore, we provide a reflection of our aim to develop a tool by rapid prototyping, whereby practitioner feedback enables concurrent maturation of the software tool and metamodel and the idea of presenting a tool to foster acceptance and practical uptake of EA Anamnesis.

Keywords—Enterprise Architecture, Design Rationale, Design Decisions, Prototype implementation

I. INTRODUCTION

As architects create blueprints for (re-)designing buildings, enterprise architects use EA modeling languages for (re-)designing organizations [1]. By taking a holistic view on an organization EA modeling languages support organizational (re-)design, such as by static impact of change analyses [1]. Prominent examples of EA languages are the Open Group standard ArchiMate [2], and the recent OMG standard Unified Profile for DoDAF/MODAF (UPDM) [3], an UML profile for describing enterprise architecture in accordance with DoDAF/MODAF.

Yet, EA modeling languages describe the EA designs, but not the reasoning behind those designs. Critical information such as the design decisions behind the resulting models are often left implicit. This also stands for the recent motivation extension of the EA modeling language ArchiMate [2]. While the motivation extension allows for expressing stakeholder intentions, it lacks well-established decision making concepts such as selection criteria, the used decision making strategy and more.

Experience from the field of software architecture shows that leaving design rationales implicit leads to “Architectural Knowledge vaporization” (cf. [4]). This means that, without design rationale, design criteria and reasons that lead to a specific design are not clear. Also, alternatives that were considered during the design process are not captured.

Among others, a lack of transparency regarding design decisions can cause design integrity issues when architects want to maintain or change the current design [5]. This means that due to a lack of the insight of the rationale, new designs are constructed in an adhoc manner, without taking into consideration constraints implied by past design decisions. Furthermore, a survey on EA rationalization amongst EA practitioners [6] suggests the relevance of architectural rationalization for motivating design decisions, and for architectural maintenance. However, the same survey shows that practitioners often forego the use of a structured template/approach when rationalizing an architecture. Instead, they capture decision characteristics in an ad hoc manner, and do so largely in plain text.

Our earlier work [7], [8], [9] proposes the EA Anamnesis approach for rationalizing EA designs. EA Anamnesis derives from the ancient greek word ανάμνησις (anáninesis), which denotes memory and repair of forgetfulness. EA Anamnesis captures design decisions made in the context of enterprise transformations and makes explicit their important characteristics, such as decision criteria, decision making strategies etc. Furthermore, the EA Anamnesis metamodel makes explicit how different design decisions are interrelated and how these decisions are linked to EA artifacts, thus allowing for a bridge between languages for EA design (prominently ArchiMate) and the corresponding design rationale.

Thus far, EA Anamnesis lacks software tool support. EA Anamnesis consists of a formal metamodel only supplemented by a way of using it. Yet, such a tool support is relevant (1) to provide for a computational assessment of EA Anamnesis. This means that we should provide evidence that the metamodel and corresponding concrete syntax (see Sect. III-B) can indeed
be implemented in a software tool. (2) to foster the practical uptake of a modeling language. As discussed by [10], amongst others, practical uptake of a modeling language is partially dependent on the modeling support provided in terms of software tools. (3) to process practitioner feedback for further tool development. Here, the idea is that the software tool is showcased as a proof-of-concept, so practitioners can react to the presented tooling support in terms of, for example, usefulness of EA Anamnesis concepts, and missing concepts and/or functionality.

As a response, this paper discusses the lessons learned during the implementation of our approach into a software prototype. More specifically, we focus our discussion on the rapid application development approach we followed for the realization of the software prototype and how this methodology helped us to further improve our metamodel by means of metamodel modifications and OCL constrains. For an elaborate discussion of the implemented metamodel see [7], [8], [9].

This paper is structured as follows: Section II introduces the metamodel of the EA Anamnesis approach. Section III discusses tool functionality, implementation aims, and illustrates its use. Section IV discussed lessons learned. Section V concludes.

II. THE EA ANAMNESIS APPROACH

In this section we introduce an insurance case study and we subsequently use it to briefly present the concepts of our integrated EA Anamnesis metamodel. The case study and the description of the concepts are accompanied by Figure 4 which depicts how these concepts are related. For a detailed description of the concepts please refer to [7], [8].

A. Case study

ArchiSurance is an insurance company that sells car insurance products using a direct-to-customer sales model. It does so to reduce its operations and product costs. Although disintermediation reduces operational costs, it also increases the risk of adverse risk profiles [11] and incomplete or faulty risk profiles of customers. These adverse profiles lead insurance companies to calculate unsuitable premiums or, even worse, to wrongfully issue insurances to customers. As a result, ArchiSurance decides to use intermediaries to sell its insurance products. After all, compiling accurate risk profiles is part of the core business of an intermediary [11].

In our scenario, an external architect called John was hired by ArchiSurance to help guide the change to an intermediary sales model. John uses ArchiMate to capture the impacts that selling insurance via an intermediary has in terms of business processes, IT infrastructure and more.

The resulting ArchiMate model is depicted in Figure 1. The initial ArchiMate model (before the transformation) is left out because of space limitations. Please refer to our previous work [8] for this EA model. We see for example how a (new) business process “customer profile registration”, owned by the insurance broker (ownership being indicated by a line between the broker and the business process), is supported by the IT applications “customer administration service intermediary” and “customer administration service ArchiSurance”.

For illustration purposes we will focus on the translation of the new business process “Customer profile registration” to EA artifacts in the application layer.

Figure 2 presents the EA Anamnesis metamodel as discussed in [7], [8]. With this metamodel we allow for (1) contextualizing the decision making process of a single decision in terms of cross cutting/intertwining decision relationships, and (2) a comparison of decision outcomes to the original decision making process.

B. The EA Anamnesis metamodel

John, uses our framework to capture EA Decisions, namely design decisions that are made in the context of an Enterprise Architecture change. The new business process “customer profile registration” has to be supported by an appropriate EA artifact (application) in the application layer of the Enterprise. This creates a new EA Issue that should be addressed with an appropriate EA design decision.

John starts a decision making process and defines the criteria that the new application should satisfy. He considers “usability”, “interoperability” and “scalability” as the most important criteria. Based on these criteria and their values he identifies four alternatives to choose from: three alternative Commercial Off-The-Shelf (COTS) applications and one alternative to upgrade the existing application in house. John is also informed about a budget limitation for the acquisition of new IT Systems. On the one hand he is aware of the quality characteristics of his alternatives, and on the other hand he has to compromise with the budget limitation. John uses EA Anamnesis to capture the strategy rationale of his decision making process based on this constraint.

EA Anamnesis also interrelates EA decisions and their consequences across the enterprise. Using translation relationships, John captures the origin of the EA decision
“Acquisition of COTS application B” (D10) which lies in the EA issue “Find an appropriate application to interface with the intermediary” (IS05), that is linked to the EA artifact “Customer administration application”. On the other hand, a decomposition relationship captures how EA Decisions are decomposed in more detailed design decisions. The design decision for a specific application user interface is related to the EA decision “Acquisition of COTS application B” (D10). Finally, John captures possible observed impacts of his implemented EA Decisions, as well as newer EA Decisions that were made to address these observed impacts through the substitution relationship.

III. PROTOTYPE IMPLEMENTATION

This section discusses our prototype development and development aims (in Sect. III-A). Furthermore, in Sect. III-B we briefly illustrate tool usage with the aid of the Archisurance case from Sect. II.

A. Prototype development and aims

Software tool functionality. We implemented the EA Anamnesis metamodel and the corresponding visualization in a software tool. This tool conforms exactly to our metamodel. No elements have been added, modified, or removed. Our software tool allows architects to rationalize architectures through a Graphical User Interface, in accordance with the EA Anamnesis metamodel. Furthermore, the software tool can export instantiations of the EA Anamnesis metamodel to a machine-interpretable, XML-based, output. In so doing, our software tool allows architects to rationalize architectures through an accessible interface and export it for further processing, hiding at the same time the technicalities of the EA Anamnesis metamodel.

Objectives and development environment. We have three key aims for developing tool support: (1) to provide for a computational assessment of EA Anamnesis. Here, we aim at testing to what extent the metamodel and corresponding concrete syntax (see Sect. III-B) can indeed be implemented in a software tool. Furthermore, the computational assessment forces one to be specific about the metamodel, thus possibly leading to metamodel changes. (2) to showcase (rudimentary) software tool support to practitioners as a means to demonstrate implementability of EA Anamnesis. We consider this relevant since tool support fosters the practical uptake of a modeling language [10]. (3) to process practitioner feedback for further tool development, during upcoming practical validation. Here, we aim at showing the tool as a proof-of-concept during case studies, so that practitioners can react to the presented tool support in terms of, for example, usefulness of EA Anamnesis concepts, and missing concepts and/or functionality. Subsequently, practitioner feedback can be processed concurrently in the metamodel and software tool.

Following objectives (1) and (3) a key requirement is the ability to develop our software tool by rapid prototyping, so that practitioner feedback and metamodel amendments can be processed without the need for extensive coding. The Microsoft Visual Studio 2013 DSL environment, which we used for tool development, allows for such rapid prototyping. It generates a model editor from a specified EA Anamnesis metamodel and corresponding concrete syntax. No further coding is required.

Figure 3 presents a sample of the development environment during tool development. On the left, we can see the domain properties of the concept of EA Decision (Name:String, State:String), and the relation of EA decision with other concepts of our approach, including relevant cardinalities (Observed Impact, EA issue, etcetera). Furthermore, Figure 3 shows how we relate the metamodel concepts presented on the left to a visualization thereof on the right. For example, the relation EADecisionReferencesTargets - which denotes substi-
The source files of our prototype implementation as well as the instantiation presented in the next section can be downloaded from: http://www.plataniotis.eu/research/ea-Anamnesis-prototype.zip.

Obviously, in the long run we aim at coding a well-structured mature tool to replace our “quick-and-dirty” prototype. However, for our current aims the prototype is sufficient.

B. Illustrative tooling scenario

Using the Archisurance scenario from Sect. II we now illustrate how our prototype implementation helps with analyzing EA design rationales.

At this point we should mention that we focus on illustrating how the prototype supports the EA Anamnesis approach in terms of a visual, computational representation of the design rationale. For an extensive illustration of EA Anamnesis, see [8].

Recall from Sect. II that John, Archisurance’s enterprise architect, used ArchiMate for modeling the to-be architecture of Archisurance. In addition, he used the EA Anamnesis tool to capture the rationale behind the to-be EA design.

Figure 4 presents the graphical instantiation of our metamodel derived from our prototype tool and provides the rationalization of the EA design of Figure 1.

For the sake of example, we concentrate on the properties and different relationships of EA decision “Acquisition of COTS application B” (D05). Table I summarizes the rationalization information provided by our approach for EA decision “Acquisition of COTS application B” (D05). Note that the table is presented for the sake of clarity: at this point we lack a visualization of the strings accompanying the various decision elements. For example for D05 we lack a tooltip “Acquisition of COTS application B”.

Returning to our scenario, we consider two translation relationships visualized by the tool: (1) between “have fitting application interface” (IS01) and “Customer administration intermediary application service A’ (D02), and (2) “find an appropriate application to interface with the intermediary” (IS02) and between “Acquisition of COTS application B” (D05). These translation relationships reflect how the requirements for a new business process were translated by John to concrete decisions in the application layer.

Furthermore, the tool visualizes the decision making process for the decision “Acquisition of COTS application B” (D05) as well as the rejected alternatives. As we can see “Acquisition of COTS application B” (D05) was selected among the alternatives “ Upgrade app” (D03) and “COTS app A” (D04) because of its highest score (w=49) during the evaluation process. For the evaluation of the alternatives a Weighted additive (WADD) decision making strategy was used to account for the difference in importance score between “usability” ($W_{usability} = 2$) and “interoperability” ($W_{interoperability} = 5$).

Furthermore, the tool represents the observed impact “Degraded user experience in the application use” (OII) of EA Decision “Acquisition of COTS application B” (D05). We can now observe that users of the customer administration application experienced difficulties to use the new application system. Thus, we see that the EA Decision “Acquisition of COTS application B” (D05) had a negative observed impact on the business process “Customer profile registration”. Moreover, we can observe that the observed impact “Degraded user experience in the application use” (OII) triggered a new issue “Find proper user interface (UI)” (IS03). In order to address the application layer issue “Find proper user interface (UI)” (IS03) John made a new Decision “user interface similar to the old one” (D06) in the application layer which is visualized by a translation relationship. EA Decision “user interface similar to the old one” (D06) substitutes Decision “Acquisition of COTS application B” (D05). This is represented by a substitution relationship.

The visualization of Figure 4 assists enterprise architects (1) to better structure and justify their decisions during the design process and (2) to analyze existing enterprise architecture designs.

IV. LESSONS LEARNED

This subsection presents lessons learned during implementation of our approach into a prototype. Particularly, we show how these lessons impact our EA Anamnesis metamodel (Figure 2).

---

TABLE I. EA DECISION 05 SUMMARY

<table>
<thead>
<tr>
<th>Title:</th>
<th>Acquisition of COTS application B</th>
</tr>
</thead>
<tbody>
<tr>
<td>EA issue:</td>
<td>EA Issue 02: Current version of customer administration application isn’t capable to support maintenance and customers administration of intermediaries application service</td>
</tr>
<tr>
<td>Decision Maker:</td>
<td>John</td>
</tr>
<tr>
<td>Layer:</td>
<td>Application</td>
</tr>
<tr>
<td>Relationships:</td>
<td>D02: introduce application service A D06: introduce interface similar to the old one</td>
</tr>
<tr>
<td>Alternatives:</td>
<td>COTS application A Upgrade existing application</td>
</tr>
<tr>
<td>Criteria:</td>
<td>usability, interoperability</td>
</tr>
<tr>
<td>Observed Impact:</td>
<td>Observed Impact 01: Reduced performance of customer registration service business process</td>
</tr>
</tbody>
</table>

---

2Requires a (trial) version of Visual Studio.
Lesson 1: EA Anamnesis currently cannot relate one decision making strategy, and its corresponding weights and criteria to multiple decisions. We attach decision making strategies to executed decisions, so that one can observe how a decision has been taken. However, in doing so each decision has a separate decision making strategy. This leads to redundancy. Whereas the same decision making strategy should be attached to both alternatives and executed decisions, for each decision we now have duplicates of the same strategy and criteria. The metamodel should be modified so that the decision making strategy information should be captured once and applied to each of the participating decisions. Figure 5a depicts the modification of the metamodel regarding this issue.

Lesson 2: A substitution relationship is a special relationship type that relates one or more EA Decisions. One of the key ideas of EA Anamnesis is to have different relationship types, so that decision elements that cut across various perspectives of the enterprise (business, application, IT infrastructure) can be related to each other in different ways [8]. Currently such relationship types always exist between an “EA decision” and an “EA issue”. This is in line with most of our purposes, except for one specific relationship type: Substitution. This is because, with substitution, we want to indicate that one decision substitutes for another. While implementing the tool, we could not represent this. Thus, the metamodel should be modified to allow direct substitution relationships between EA Decisions. Figure 5b depicts the required modification of the metamodel.

Lesson 3: A relationship type between executed Decision1 and Issue1 determines how EA Issue1 is related with an executed Decision2. According to the metamodel, in order to create a translation/decomposition relationship between two EA Decisions, we have to create two separate relationships of this type: one for EADecision-EAIssue and one for EAIssue-EADecision.

However, the relationship EAIssue1-EADecision2 should be automatically determined by the relationship EADecision1-EAIssue1. For example: if a translation relation exists between executed EADecision1 and EAIssue1, this automatically implies that a translation relation exists between EAIssue1 and executed EADecision2.

Similarly an alternative relationship can be also automatically determined by the rejected state of an EA Decision. For example: if EADecision3 has a rejected state and is related with EAIssue1, the relationship between EADecision3 and EAIssue1 should automatically be set to “alternative”. To address these issues, we introduce bi-directional relationships in our metamodel and complement them with Constraint Rules defined in Object Constraint Language (OCL). These constraints are presented in Figure 6.

Fig. 4. Prototype tool visualization

Fig. 5. Modifications of the EA Anamnesis metamodel for lesson 1 (a) and lesson 2 (b).
of doing this is by evaluating the actual practical usefulness of the concepts of the decision making strategy viewpoint. For example we capture the strategy rationale for selecting a decision making strategy, but whether the effort for capturing this outweighs the received benefits remains to be seen.

ACKNOWLEDGMENTS.

This work has been partially sponsored by the Fonds National de la Recherche Luxembourg (www.fnr.lu), via the PEARL programme.

REFERENCES