Relating Decisions in Enterprise Architecture using Decision Design Graphs

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
Public Research Centre Henri Tudor,
Luxembourg, Luxembourg
EE-Team, Luxembourg, Luxembourg
sybren.dekinderen@tudor.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, such as ArchiMate, describe an enterprise holistically. In doing so, they show an enterprise's business products and services, and how these are realized by IT infrastructure and applications. However, EA modeling languages lack the capability to capture design rationales for decisions that lead to specific architectural designs. In our previous work we presented the EA Anamnesis approach for capturing decision details behind EA models. In doing so, we focused on capturing individual architectural decisions (in terms of alternatives, decision criteria, et cetera).**

In this paper we present an approach for relating architectural decisions. Using decision design graphs, we make explicit how decisions from different enterprise domains (Business, Application, Technology) relate to each other. For example, how decisions taken on a business level affect IT decisions and vice versa. Our approach is inspired by well-known mechanisms for capturing architectural rationales in software architecture.

Specifically we contribute: (1) a decision relationship metamodel for enterprise architecture, with a focus on recording the impact of decisions (2) the notion of a Decision Design Graph for enterprise architecture, a visual representation of this metamodel, and (3) an illustrative example illustrating the potential usefulness of capturing decision relationships.

Keywords*—*Enterprise Architecture, Decision relationships, Design Rationale, Decision Capturing, Decision Design Graph

I. Introduction

Enterprise Architecture (EA) languages, such as the Open Group Standard ArchiMate [1], are recognized for connecting an organization's IT infrastructure and applications to the business processes they support and the products/services that are in turn realized by the business processes [2], [3]. The resulting holistic perspective on an enterprise helps clarify the business advantages of IT, analyze cost structures and more [4].

However, while Enterprise Architecture modeling languages allow for modeling an enterprise holistically, the design decisions behind the resulting models are often left implicit. As we point out in [5], software architecture literature shows that leaving design rationales implicit leads to 'Architectural Knowledge vaporization' (cf. [6]). This means that, without a 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.

Such a lack of design transparency implies that new designs are constructed in an ad hoc manner, without taking into consideration constraints implied by past design decisions [7]. Moreover one loses track of lessons learned in terms of anticipated and, importantly, *unanticipated* consequences of design decisions [8].

The need for explicit architectural rationalization is further compounded by a survey amongst software architects on recording design rationales [9]. Here a large majority of architects (85,1%) underline the importance of architectural rationalization. Another interesting finding of this survey was that architects declared that after some time they frequently forget their own decisions. Furthermore, anecdotal evidence from six exploratory interviews we conducted with senior enterprise architects suggests that Architects are often external consultants. Without architectural rationalization, namely, a new architect has to search through architectural designs and unstructured information of requirements to understand and analyze the architecture.

In our earlier work [10], [5] we introduce an approach for the rationalization of enterprise architectures by capturing EA design decision details, and the EA decision making process. We refer to this approach as EA Anamnesis, from the ancient greek work $\alpha\nu\acute{\alpha}\mu\nu\eta\sigma\iota\varsigma$ (/ˌænæmˈniːsɪs/), which denotes memory and repair of forgetfulness. The metamodel is grounded in Decision Representation Language (DRL) [11] applications in software architecture. Moreover, it is based on well-established decision making literature [12], [13], [14], [15], [16]. For now, the EA Anamnesis approach is focused on *individual* decisions, capturing for each characteristics such as the issue leading up the decision, the considered alternatives, decision-making criteria, et cetera.

In this paper we extend the EA Anamnesis approach with a metamodel for capturing *relationships between decisions*. This metamodel enables the capturing and visualization of the impact of decisions across the enterprise. We consider this as a useful extension because decisions rarely exist in isolation [17], [18], [7]. Rather, decisions are often cross cutting and intertwined [6]: a decision made on a business

The Enterprise Engineering Team (EE-Team) is a collaboration between Public Research Centre Henri Tudor, Radboud University and HAN University of Applied Sciences (www.ee-team.eu)

level can have (un-)anticipated consequences on an IT-level and vice versa.

In particular, our contribution is threefold. First, we contribute a metamodel for capturing decision relationships. Here, the focus is on the ex-post capturing of decision impacts. Second, we provide a visualization of the decision relationship metamodel in terms of Decision Design Graphs. Both the metamodel and Decision Design Graph are grounded in well-established software engineering approaches for capturing decision relationships [17], [18], [7]. Third, with an illustrative scenario, we contribute an illustration of the potential usefulness of considering decision relationships in enterprise architecture.

Note that existing rationalization approaches from software architecture, such as [17], [18], [7], do not consider business issues, such as decisions related to business processes. Furthermore, while the EA language ArchiMate 2.0 [1], [19] has a motivational layer, it lacks concepts important for rationalization such as considered alternatives, decision criteria et cetera. As such, ArchiMate 2.0 is not a suitable language for architectural rationalization.

In section II, we introduce our metamodel and a fictitious insurance case that is used for illustration. Subsequently, section III introduces Decision Design Graphs (which are based on our metamodel), and presents a scenario that illustrates the usefulness of considering decision relationships. Finally, sections IV and V discuss related work and present conclusions, respectively.

II. EA DECISIONS RELATIONSHIPS

In this section we introduce an insurance case study (section II-A), and subsequently use it to illustrate our EA decision relationships metamodel (section II-B). Note that the insurance case study is fictitious yet realistic. This is because it is based on the running case study used to illustrate the ArchiMate specification [1], which in turn is based on a real insurance company.

A. Case study: ArchiSurance

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.

Figure 1 presents the partial (Business and Application layers) ArchiSurance direct-to-customer sales model, modeled with the EA modeling language ArchiMate. Two business services support the sales model of ArchiSurance: “Car insurance registration service” and “Car insurance service”. ArchiMate helps us to understand the dependencies between different perspectives on an enterprise. For example, in Figure 1 we see that the business service “Car insurance registration service” is realized by a business process “Register customer profile”. In turn, we also see that this business process is supported by the application service “Customer administration service”.

Although disintermediation reduces operational costs it also increases the risk of adverse risk profiles [20], 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 response, 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 [20].

In our scenario, an external architect called *John* was hired by ArchiSurance to help guide the change to an intermediary sales model.

Scenario:

John uses ArchiMate to capture the impacts that selling insurance via an intermediary has in terms of business processes, IT infrastructure and more. For illustration purposes we will focus on the translation of the new business process “Customer profile registration” to EA artifacts in the application layer. The resulting ArchiMate model is depicted in Figure 2.

Here 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”.

B. EA decision relationships metamodel

In this section we present the EA decision relationship metamodel, presented in Figure 3. The concepts of this metamodel are grounded in established literature for capturing design rationale in software architectures [17], [18], [7], [21]. We apply the concepts in the area of enterprise architecture instead.

The metamodel focuses on (1) representing the different relationships of captured EA decisions, (2) grouping EA decisions according to the enterprise layer and artifacts they belong to and (3) representing the observed impact of these decisions on the enterprise architecture. Note that the examples here are

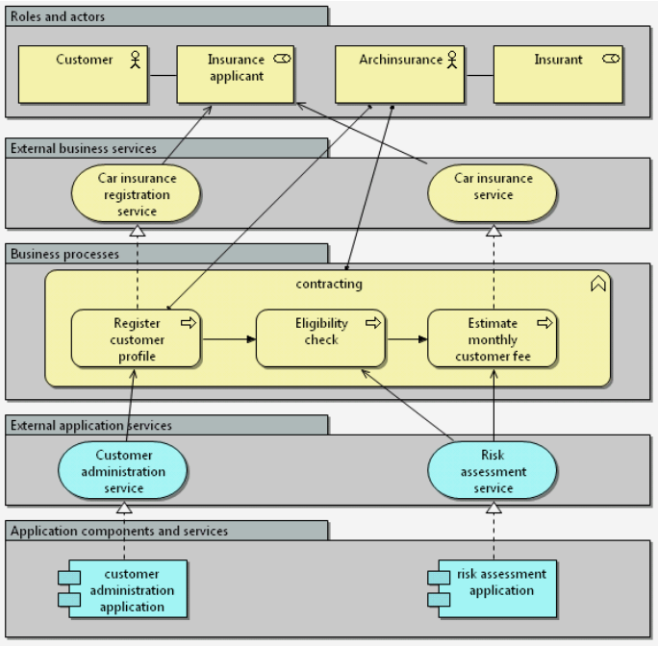

Fig. 1. ArchiSurance direct-to-customer EA model

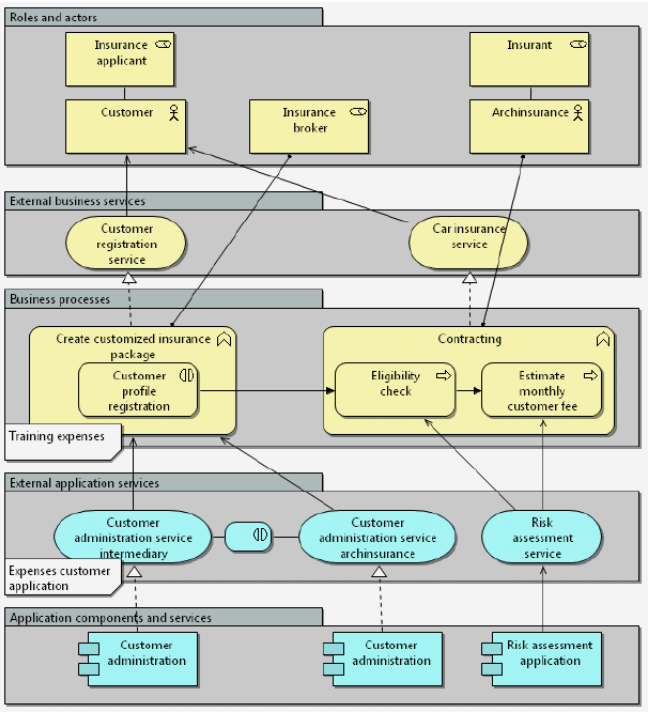

Fig. 2. ArchiSurance intermediary EA model

related in the decision decision design graph from Figure 4, an instrument that we detail further in the next section.

EA decision:

An EA decision shows decisions that are made and captured in the context of an Enterprise transformation [8].

Example: John makes the EA decision 'make customer profile registration via intermediary'.

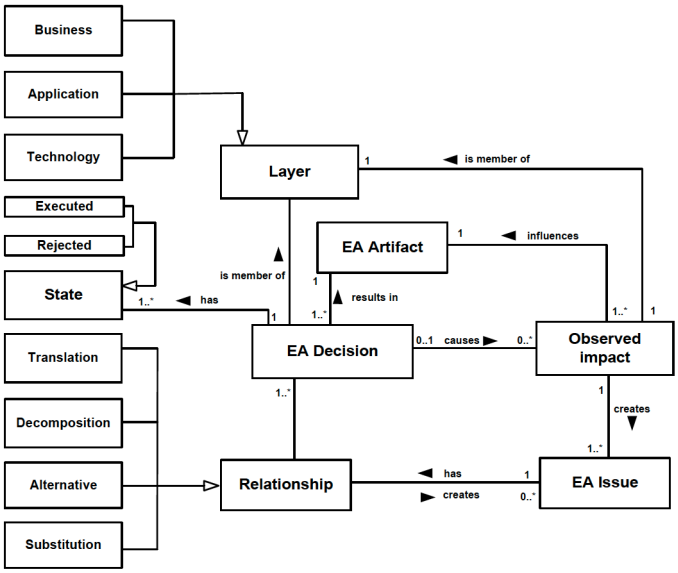

Fig. 3. EA decisions relationship metamodel

EA issue:

Similar to the concept of an issue from [18], an EA issue represents the architectural design problem that enterprise architects have to address during the Enterprise transformation process.

Example: The EA issue 'create an appropriate application service to support new business process' resulting from the EA decision 'introduce a new business process for customer profile registration'.

EA artifact:

An EA artifact (similar to concept of an architecture element [7]) is either the direct result produced from a set of executed EA decisions, or a representation of this result. For now, we use an EA artifact to refer to architectural representations. Specifically, we use it as a bridging concept towards the EA modeling language ArchiMate, whereby an EA artifact allows us to link EA decisions to concepts from ArchiMate.

Example: The EA artifact 'Customer administration application' in the ArchiMate model in Figure 2 is linked to, amongst others, the EA decision 'acquisition of COTS application B"' and EA decisions "Application interface type 1" and "Application interface type 2".

Layer:

In line with the ArchiMate language [1] an enterprise is specified in three layers: *Business, Application and Technology*. Using these three layers, we express an enterprise *holistically*, showing not only applications and physical IT infrastructure (expressed through the application and technology layers), but also how an enterprise's IT impacts/is impacted by an enterprise's products and services and its business strategy and processes.

Example: The EA decision 01 'make customer profile registration via intermediary' (Figure 4) is a member of the business layer of Archisurance.

State:

An EA decision can be in an executed or rejected state [21]. In an executed state an EA decision has already been made and executed. A rejected decision, on the other hand, is a decision that was considered as an alternative during the decision making process but was rejected because another decision was more appropriate.

Example: John had to address EA issue 05 'find an appropriate application to interface with the intermediary'. 'Acquisition of COTS application B'(D10) is the executed decision, whereas decisions 'COTS application A'(D08) and 'upgrade existing application (inhouse)' (D09) are the rejected (alternative) decisions.

Relationship:

The role of relationship concepts is to make the different types of relationships between EA decisions explicit. Based on ontologies for software architecture design decisions [17], [21], we define four types of relationships:

- **Translation relationship:**
  Translation relationships illustrate relationships between decisions/EA issues that belong on different layers/EA artifacts. During the enterprise transformation process architects translate the requirements that new EA artifacts impose (EA issue) to decisions that will support these requirements by means of another EA artifact [22].

  Example: The EA decision 'make customer profile registration via intermediary' translates to the issue 'find an appropriate application service'. Subsequently this issue translates to a second EA decision 'introduce application service A'.

- **Decomposition relationship:**
  The Decomposition relationship is in line with 'Comprises (Is Made of, Decomposes into)' of Kruchten's ontology [21]. Decomposition relationships signify how generic EA decisions decompose into more detailed design decisions.

  Example: The EA decision 'acquisition of COTS application B' has a decomposition relationship with EA decision 'Application interface type 1'.

- **Alternative relationship:**
  This relationship type [21] illustrates the EA decisions that were rejected (alternatives) in order to address a specific EA issue.

  Example: Rejected EA decisions 'COTS application A'(D08) and 'Upgrade existing application (inhouse)' (D09) have an alternative relationship with EA issue 05 'find an appropriate application to interface with the intermediary'. This signifies that these decisions were the alternatives for this issue.

- **Substitution relationship:** A substitution relationship explicates how one EA decision replaces another EA decision. An EA decision can be replaced when it creates a negative impact in the enterprise architecture.

  Example: The EA decision 'Application interface 1' has a negative observed impact on the business process 'Customer profile registration'. This is because it leads users to make mistakes, as we will see with the concept 'observed impact'. As such, it is replaced by the EA decision 'Application interface 2'.

Observed impact:

The observed impact concept signifies an *unanticipated* consequence of an already made decision to an EA artifact. This is opposed anticipated consequences, as indicated by relationships such as translation or decomposition. Observed impacts can be positive or negative.

In current everyday practice, architects model *anticipated* consequences using what-if-scenarios [4]. Unfortunately, not every possible impact of made EA decisions can be predicted. This is especially true for enterprise architecture, where one considers impacts across the enterprise rather than in one specific (e.g. technical) part. The falsiability of EA decisions is challenged in the a-posteriori analysis of the architecture [8]. Some of the consequences of EA decisions are revealed during the implementation phase, or during the maintenance of the existing architecture design. These unanticipated consequences are captured exactly by the concept of an observed impact.

For us the main usefulness of capturing observed impacts is that they can be used by architects to avoid decisions with negative consequences in future designs of the architecture.

Causes, Creates relationships for Observed impact:

EA decisions *cause* one or more observed impacts. Furthermore, these observed impacts *create* new EA issues.

Example: The EA decision 'Application interface type 1" (D11) has an observed impact 'degraded user experience in the application use'.

III. Visualizing EA decision relationships

In this section we introduce our decision relationships visualization in terms of Decision Design Graphs (in section III-A). Subsequently, in section III-B, we illustrate the potential usefulness of the graph with the ArchiSurance scenario from section II-A.

A. Decision Design Graphs

As stated Decision Design Graphs complement our previously introduced EA Anamnesis approach to consider relationships between decisions.

We consider EA Anamnesis useful for capturing *individual* decisions. However, the information captured with EA Anamnesis provides only a taxonomy of textual information per decision. Consider, for example, the EA Anamnesis rationalization of the EA decision "Acquisition of COTS application B", of our running insurance case. Here we see that, for one, the EA issue field in Table I only provides an explanation of the problem in a textual format. It lacks explication of the relationships of this decision with other decisions, the issues caused by a decision, or the observed impacts of a decision.

To remedy this, we capture EA decision relationships with our metamodel and subsequently visualize them with a Decision Design Graph (DDG).

DDGs are used in software architecture to visualize architectural rationalization [23], but we use them for rationalization of enterprise architecture decisions.

B. Using Decision Design Graphs to capture EA decisions relationships

Using our ArchiSurance scenario, we now show how DDGs extend the EA Anamnesis approach by making explicit the relationships of Enterprise architecture issues, decisions and their observed impacts. In accordance with the focus on decision relationships and decision consequences we do not elaborate on other decision details such as selection criteria and factors that affect the decisions making process, see our previous work [5], [10].

Scenario. Our external architect John has coordinated the transformation into the intermediary sales model depicted in Figure 2. He translated the requirements imposed by the introduction of an insurance broker into concrete EA artifacts. For the implementation of these EA artifacts a number of EA decisions was made. John, in parallel with ArchiMate modeling language, has used our approach to capture the relationships of decisions and their impacts across the enterprise.

Figure 4 depicts the DDG of EA decisions and their relationships, captured by John during the enterprise transformation of ArchiSurance.

Let us assume that a newly hired Enterprise Architect, Bob, wants to understand the Enterprise Architecture of the organization by using EA Anamnesis, taking into consideration also EA decision relationships. For the sake of example, we concentrate on a snapshot of 13 EA decisions. Using the DDG (Figure 4) in addition to the EA Anamnesis table, Bob can understand now understand the following.

*Using **translation relationships**, Bob can understand the origin of the EA decision "Acquisition of COTS application B" (D10), which is linked to the EA artifact "Customer administration application". Note that the link between EA decision and EA artifact is formalized through the metamodel relation (EA decision)'results in'(EA artifact) in Figure 3.*

First, Bob finds a relationship between the EA decision "Acquisition of COTS application B" (D10) and the EA issue "Find an appropriate application to interface with the intermediary"(IS05). Furthermore he finds that the issue "Find an appropriate application to interface with the intermediary" (IS05) has a translation relationship with "Introduce application service A" (D06).

Subsequently, using translation relationships, Bob can trace "Introduce application service A" (D06) to "Make customer profile registration via intermediary" (D01), via the EA issue "'Create an appropriate application service to support new business process" (IS03) (see Figure 4).

TABLE I. EA DECISION 10 DETAILS

Title:	Acquisition of COTS application B
EA issue:	EA Issue 05: Current version of customer administration application isn't capable to support maintenance and customers administration of intermediaries application service
Decision Maker:	John
Layer:	Application
Alternatives:	COTS application A Upgrade existing application (inhouse)
Rationale:	Scalability: Application is ready to support new application services
Criteria:	Customized reports capability, interoperability, scalability, cost
Observed Impact:	Observed Impact 01: Reduced performance of customer registration service business process

During the ex-post analysis of the EA design, the captured relationships of translations of EA decisions can create a chain. This chain is useful for the traceability of the decision making process across the enterprise. In particular, it allows for tracing technical decisions to business decisions and vice versa. EA stakeholders of different layers of the enterprise (for ArchiMate, the layers are Business, Application and Technology) can identify the set of EA decisions that comprise an architectural artifact. Also, translation relationships can identify how these decisions affect decisions in other layers of the enterprise.

*After translation analysis, Bob can inspect **alternative relationships** between the EA issue "Finding an appropriate application" (IS05) on the one hand, and EA decisions "COTS application A" (D08) and "Upgrade existing application (inhouse)" (D09) on the other hand. Here, the alternative relationship signifies that D08 and D09 were alternative decisions for issue "finding an appropriate application".*

Note here that the EA Anamnesis approach provide rationalization information about the selection of individual EA decisions over alternatives. For example: Table I details the rationalization for selecting D10. This is detailed further in [10].

*Next, let us assume that Bob wants to examine how EA decision 10 "Acquisition of COTS application B" was decomposed to a set of specific EA decisions. The **decomposition relationship** between EA decision 10 "Acquisition of COTS application B" and decision 11 "Application interface type 1" signifies that EA decision 11 is a sub decision of EA decision 10.*

*Next, Bob wants to determine which decisions constitute the implementation of a specific **EA artifact**. EA decisions, enclosed by the dashed circle line, comprise a specific EA artifact. For the EA artifact "Customer administration application" EA decisions "Acquisition of COTS application B"(D10), "Application interface type 1"(D11) and "Application interface type 2" (D13) were executed.*

*Next Bob inspects the **observed impacts**, the unexpected consequences of a decision. Consider the business process "Customer profile registration" (in the ArchiMate model in Figure 2, and as an EA artifact in Figure 4). For this business process, decisions have been taken such as the top level decision 01 "Make customer profile registration via intermediary", leading to the issue 03 "'Create an appropriate application service to support new business process" (See scenario so far). As a result of these decisions, the users of this business process had to use the application "Customer administration application" with a new application interface. This new interface is implied by the decision "Application interface type 1"(D11), a sub decision of "Acquisition of COTS application B" (D10).*

Using the concept of an Observed impact, Bob can now observe that users of "Customer administration application" had difficulties using this new application interface. This is signified by the observed impact 01 "Degraded user experience in the application use" (OI1 in Figure 4). As such, Bob finds

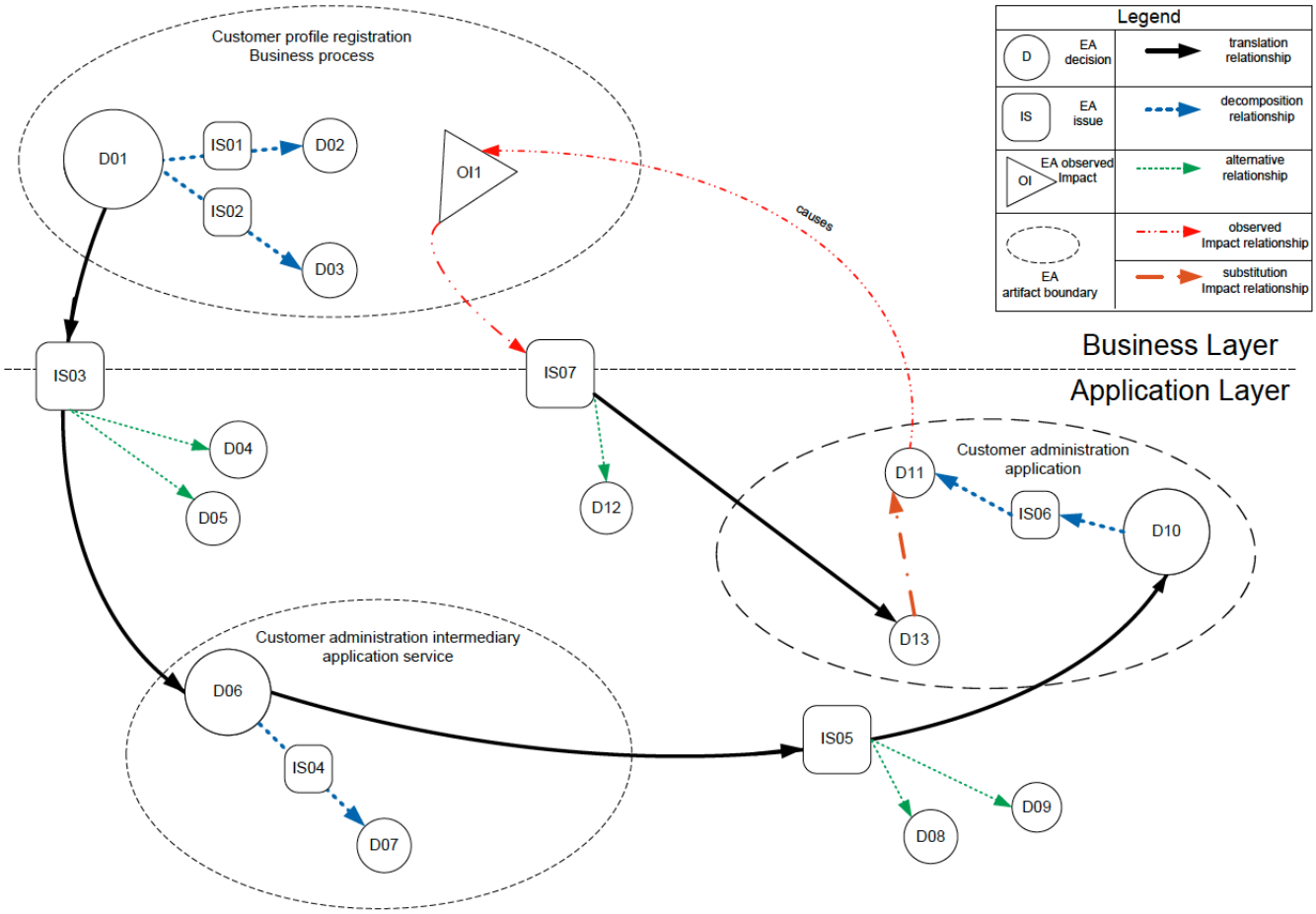

Fig. 4. EA decisions relationships visualization

that EA decision 11 "Application interface 1" had a negative observed impact on the business process "Customer profile registration".

Next Bob sees that the observed impact "Degraded user experience in the application use" leads to the EA issue 07 "have fitting application interface". In turn, he notices that there are two possible decisions for this problem (1) "Training of users on the new application" (EA decision 12) and (2) "replace of existing application interface with an interface similar to the old one" (EA decision 13). Finally, Bob notices that his predecessor John decided for the second option (the executed EA decision 13) "Application interface type 2" and rejected the first one (the rejected EA decision 12).

The role of the observed impact relationship is to capture unanticipated consequences of decisions. By examining observed impacts, architects become aware of possible impacts that arise from EA decisions.

Finally, Bob can inspect the ***substitution relationship****. In see Figure 4, we see a relation between EA decision 13 "Application interface type 2" and EA decision 11 "Application interface type 1". This signifies that decision 13 was executed and that it replaced decision 11 that caused the problem.*

As a final part of our scenario, we now skip two years ahead in time to illustrate how the captured decisions can aid in decision making.

Two years later: Archisurance achieved its strategic goal with the addition of intermediaries. Customer profiles are better calculated by intermediaries. Archisurance management decides to adopt the same strategy for the remainder of its insurance products. Bob is called to translate this strategic goal in an appropriate enterprise architecture design. He uses ArchiMate to design the "to be" architecture. Our approach helps Bob to identify similar EA issues faced by his predecessor John, and the decisions taken as a response. By searching for similar EA issues from the past, Bob can identify what kind of decisions were taken to address issues. Moreover, he can observe expected and unexpected outcomes. If the outcome is successful, Bob can reuse some of these good practices. If the outcome is negative, Bob can avoid these decisions or anticipate on problems he may face. For example, if Bob has to decide for an appropriate Application interface (similar to EA issue 06), our approach advises Bob of the possibility of a problem in the new business process. Bob can apply EA decision 13 "Application interface type 2", which is the interface that user are familiar with, without again repeating the decision with the negative outcome. Alternatively he can inform management of this potential problem. A training of future users can be organized before the new application

interface is in place.

IV. RELATED WORK

Several design rationale approaches have been developed for the domain of software architecture. Two main categories of approaches exist: template based and model based. Template based approaches, such as [18], [24], capture in textual format important details of Architectural Decisions like Rationale, Issue, Implications et cetera. Although template based approaches provide useful information about decisions, this information is unstructured.

Model based approaches [6], [7], [17], on the other hand, provide information similar to template based approaches but use dedicated decision models. These models not only provide important attributes for each decision, but also a means to relate these decisions with software architecture artifacts and other decisions. By using a structured information format for decisions and relationship information, these models provide a better architectural overview of software systems by representing the impact of these decisions to the architecture.

However, Software Architecture is only a subset of Enterprise Architecture [25], first and foremost because Software Architecture deals less with business issues.

In parallel with rationalization techniques, there are other approaches for representing decision relationships. Ran and Kuusela [26] proposed the representation of relationships between design decisions in the form of Design Decision Trees. Kruchten [21] extends this approach by incorporating additional relationship types between decisions. Our approach complements model based approaches for Software Architecture by providing attributes specific for EA decisions as well as more specific dependency and relationship types between EA decisions.

Related to the impact analysis of our approach, there are root causes analysis (RCA) techniques [27] that aim at identifing the root causes of a problem. By identifying problems and their symptoms, reappearance of problems can be prevented. However these techniques do not take into consideration how the decision maker came to a certain decision by examining different alternatives, criteria et cetera. Furthermore, RCA techniques lack a formal metamodel, and lack a link to enterprise architecture.

Finally, goal modeling techniques can be used to model the motivations for designs [28]. While goal modeling techniques usually have more expressivity for motivations than the motivational layer of ArchiMate 2.0, they are not focused on decision rationalization. As such, goal modeling techniques do not express rationalization concepts such as issues and alternatives.

V. CONCLUSION

In this paper we introduced a metamodel and corresponding visualization for capturing relationships between Enterprise Architecture decisions. This metamodel is an extension for the EA Anamnesis approach, our previous work. With this extension, we allow for capturing decision relationship dependencies, showing what (un)anticipated consequences decisions have across the enterprise. For example: how a business level decision impacts an IT level decisions and vice versa.

In this paper, we only provide a metamodel that ex-post captures the decision that has been made. However, to actually make the decision, different stakeholders with different *individual* rationales and stakes, from business as well as IT, have to coordinate to collaboratively come to the final architectural decision graph. How to collaboratively create a design decision graph, taking into consideration individual concerns is for now part of future work. As a starting point, we will look at literature on Group Decision Support Systems and Multi-Criteria Decision Analysis theories to further extend our research in Design Rationalization and Communication of Enterprise Architecture during the design process.

Last but not least, one of our major challenges is to investigate the return of capturing effort for our approach. Our design rationale assists architects to better understand existing EA designs, but the effort of capturing this information might be a dissuasive factor. To address this issue our research will focus on ways to decrease the capturing effort. One way of doing this is by evaluating the actual practical usefulness of the concepts of our approach.

ACKNOWLEDGMENTS.

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

REFERENCES

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

[2] M. Op't Land, E. Proper, M. Waage, J. Cloo, and C. Steghuis, *Enterprise architecture: creating value by informed governance*. Springer, 2008.

[3] J. Hoogervorst, "Enterprise architecture: Enabling integration, agility and change," *International Journal of Cooperative Information Systems*, vol. 13, no. 03, pp. 213–233, 2004.

[4] M. Lankhorst, *Enterprise architecture at work: Modelling, communication and analysis*. Springer, 2009.

[5] G. Plataniotis, S. de Kinderen, and H. A. Proper, "Capturing decision making strategies in enterprise architecture - a viewpoint," in *EMMSAD 2013 (Exploring Modelling Methods for Systems Analysis and Design), to appear*, 2013.

[6] A. Jansen and J. Bosch, "Software architecture as a set of architectural design decisions," in *Software Architecture, 2005. WICSA 2005. 5th Working IEEE/IFIP Conference on*. IEEE, 2005, pp. 109–120.

[7] A. Tang, Y. Jin, and J. Han, "A rationale-based architecture model for design traceability and reasoning," *Journal of Systems and Software*, vol. 80, no. 6, pp. 918 – 934, 2007. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0164121206002287

[8] H. Proper and M. Op t Land, "Lines in the water," in *Practice-Driven Research on Enterprise Transformation*, ser. Lecture Notes in Business Information Processing, F. Harmsen, E. Proper, F. Schalkwijk, J. Barjis, and S. Overbeek, Eds. Springer Berlin Heidelberg, 2010, vol. 69, pp. 193–216. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-16770-6_9

[9] A. Tang, M. A. Babar, I. Gorton, and J. Han, "A survey of architecture design rationale," *Journal of Systems and Software*, vol. 79, no. 12, pp. 1792 – 1804, 2006. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0164121206001415

[10] G. Plataniotis, S. de Kinderen, and H. A. Proper, "Ea anamnesis: towards an approach for enterprise architecture rationalization," in *Proceedings of the 2012 workshop on Domain-specific modeling*, ser. DSM '12. New York, NY, USA: ACM, 2012, pp. 27–32. [Online]. Available: http://doi.acm.org/10.1145/2420918.2420927

[11] J. Lee, "Extending the potts and bruns model for recording design rationale," in *Software Engineering, 1991. Proceedings., 13th International Conference on*. IEEE, 1991, pp. 114–125.

[12] H. Einhorn, "The use of nonlinear, noncompensatory models in decision making." *Psychological bulletin*, vol. 73, no. 3, p. 221, 1970.

[13] J. Payne, "Task complexity and contingent processing in decision making: An information search and protocol analysis," *Organizational behavior and human performance*, vol. 16, no. 2, pp. 366–387, 1976.

[14] O. Svenson, "Process descriptions of decision making," *Organizational behavior and human performance*, vol. 23, no. 1, pp. 86–112, 1979.

[15] B. Alenljung and A. Persson, "Portraying the practice of decision-making in requirements engineering: a case of large scale bespoke development," *Requirements engineering*, vol. 13, no. 4, pp. 257–279, 2008.

[16] J. Orasanu and T. Connolly, "The reinvention of decision making." 1993.

[17] P. Kruchten, "An ontology of architectural design decisions in software intensive systems," in *2nd Groningen Workshop on Software Variability*, 2004, pp. 54–61.

[18] J. Tyree and A. Akerman, "Architecture decisions: Demystifying architecture," *Software, IEEE*, vol. 22, no. 2, pp. 19–27, 2005.

[19] C. Azevedo, J. Almeida, M. Van Sinderen, D. Quartel, and G. Guizzardi, "An ontology-based semantics for the motivation extension to archimate," in *Enterprise Distributed Object Computing Conference (EDOC), 2011 15th IEEE International*, 2011, pp. 25–34.

[20] J. Cummins and N. Doherty, "The economics of insurance intermediaries," *Journal of Risk and Insurance*, vol. 73, no. 3, pp. 359–396, 2006.

[21] P. Kruchten, P. Lago, and H. Vliet, "Building up and reasoning about architectural knowledge," in *Quality of Software Architectures*, ser. Lecture Notes in Computer Science, C. Hofmeister, I. Crnkovic, and R. Reussner, Eds. Springer Berlin Heidelberg, 2006, vol. 4214, pp. 43–58.

[22] M. Op't Land and H. Proper, "Impact of principles on enterprise engineering," *ECIS 2007 Proceedings*, 2007.

[23] O. Zimmermann, J. Koehler, F. Leymann, R. Polley, and N. Schuster, "Managing architectural decision models with dependency relations, integrity constraints, and production rules," *Journal of Systems and Software*, vol. 82, no. 8, pp. 1249 – 1267, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0164121209000181

[24] J. Savolainen, "Tools for design rationale documentation in the development of a product family," in *Position Paper Proceedings of 1st Working IFIP Conference on Software Architecture, San Antonio, Texas*, 1999.

[25] C. Coggins and J. Speigel, "The methodology for business transformation v1.5: A practical approach to segment architecture," *Journal of Enterprise Architecture*, 2007.

[26] A. Ran and J. Kuusela, "Design decision trees," in *Proceedings of the 8th International Workshop on Software Specification and Design*, ser. IWSSD '96. Washington, DC, USA: IEEE Computer Society, 1996. [Online]. Available: http://dl.acm.org/citation.cfm?id=857204.858278

[27] A. Gwiazda, "Quality tools in a process of technical project management," *Journal of Achievements in Materials and Manufacturing Engineering*, vol. 18, no. 1-2, pp. 439–442, 2006.

[28] J. Horkoff and E. Yu, "Comparison and evaluation of goal-oriented satisfaction analysis techniques," *Requirements Engineering*, pp. 1–24, 2012. [Online]. Available: http://dx.doi.org/10.1007/s00766-011-0143-y