Traceability and Modeling of Requirements in Enterprise Architecture from a Design Rationale Perspective

Georgios Plataniotis, Qin Ma, Erik Proper
Luxembourg Institute of Science and Technology, Luxembourg
Radboud University Nijmegen, Nijmegen, the Netherlands
EE-Team, Luxembourg, Luxembourg
{georgios.plataniotis, qin.ma, erik.proper}list.lu

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

***Abstract*—Enterprise architecture modeling languages capture holistically the structure of an enterprise. Therefore, they represent how the services and business processes of an organization are supported by IT infrastructure and applications. However, the reasoning behind the selection of specific design decisions in the architecture remains usually implicit. Our earlier work proposes the EA Anamnesis approach for the capturing of design rationalization information. Its major contribution is a formal metamodel that captures the reasoning and the inter-relationships of design decisions. We have already evaluated our approach in a real world case where EA Anamnesis successfully captured design rationales in the context of an enterprise transformation. However an important deficiency was also observed during this evaluation, which points out the need for the traceability of enterprise architecture designs to the functional and non-functional requirements that were used for this design. This paper extend the EA Anamnesis approach to address this important aspect of EA rationalization. By doing as such we provide an explicit bridging between the problem space which is described by the different requirements and the actual solution space which is described by specific design decisions. More specifically, we contribute: (1) an enhanced EA metamodel that also supports requirements traceability and (2) an illustrative case study from the insurance sector to demonstrate the potential usefulness.**

Keywords*—*Enterprise Architecture, Design Rationale, Functional requirements, Non functional requirements, Decision making process

I. Introduction

Enterprise Architecture (EA) models are considered as an instrument to represent an enterprise holistically [1], linking perspectives on an organization which are usually considered in isolation. In doing so, one can consider the organization-wide impact of a change [1], [2], expressing its complete business-to-IT-stack [3]. For example, for a newly introduced IT application, EA models can be used to consider implications on business processes, human resources, organizational goals, and more.

Although EA models can be used to capture the holistic design of an organization, they seldom specify the design decisions behind the resulting models. Even if we should be careful with the analogy, 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, one leaves implicit design criteria, reasons for a design, and design alternatives.

As a result of lacking rationalization architects are unable to justify their designs [5]. Furthermore new designs are constructed in an ad hoc manner without taking into consideration constraints implied by past design decisions [5]. Here, constraints refer to boundaries implied by the design. These boundaries can be of business or technical nature, such as the choice for a programming language implied by choosing a particular application environment.

Moreover, a survey amongst enterprise architecture practitioners [6] provides indications of the usefulness of design rationalization for motivating design decisions, and for architectural maintenance. At the same time, however, the survey shows that practitioners often forego the use of a structured template/approach when rationalizing an architecture, relying instead on ad hoc information capturing in tools such as Microsoft Office.

In our earlier work [7], [8], [9] we introduced the EA Anamnesis approach for architectural rationalization. EA Anamnesis captures decision characteristics such as decision criteria and used decision making strategy, and shows the relation between business-level and IT-level decisions. Furthermore, EA Anamnesis relies on a metamodel-based formal linkage between EA modeling languages (mainly ArchiMate) and the corresponding decision aspects to realize the connection between EA designs and EA rationale.

So far we have already applied the EA Anamnesis approach in the context of a real world enterprise architecture transformation. The evaluation showed that EA Anamnesis could sufficiently capture design rationale and enabled practitioners to structure and replay their decision making processes. More importantly, the evaluation also indicated that we should work

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`)

further on the traceability from specific design decisions back to the original requirements. More specifically our approach already provides adequate information on how an EA design issue was addressed in the solution space by making specific design decisions but it does not explicitly interrelate these decisions with the requirements that the architect took into consideration while he was deciding.

As a response, in this paper we enhance the EA Anamnesis metamodel to provide an explicit linkage with requirements engineering concepts. By doing so we enable backward / forward traceability from design decisions to the given requirements and vice versa, and the capability to identify if the architectural decisions are aligned to the essential enterprise architecture requirements. According to [10] this capability is one of the main purposes of the enterprise architecture function in an organization. The refinement of the metamodel was based on existing approaches from the requirements engineering domain.

This paper is structured as follows. Section II presents the background literature in requirement engineering and decision making, Section III introduces the refined metamodel, while Section IV illustrates the use of our approach with an insurance industry case study. Section V presents related work from the domains of requirements engineering and design rationale and finally Section VI concludes.

II. BACKGROUND

The work reported in this paper is based on established literature in requirements engineering and architectural decision making. This section reports on the two streams that were examined: (1) decision making theories which provide insight on how people decide given specific contexts (2) and requirements engineering approaches for architectural decision making which show how the decision making is affected by the requirements.

A. Decision making and requirements engineering

Decision making strategies generally fall in two main categories: compensatory and noncompensatory [11], [12], [13], [14]. We briefly explain these strategies with a car buying example. In this example, a customer selects a car based upon the criteria 'color of car', 'carbon emission', 'small size of car' and 'gasoline consumption'.

In compensatory decision making [11], alternatives are evaluated exhaustively, taking *all* criteria and their trade-offs into consideration. Criteria with high values compensate for criteria with lower values. Finally, the alternative with the highest score is selected. For our car buying example, this implies that a customer considers all four criteria 'color of car', 'carbon emission', 'small size of car' and 'gasoline consumption'. For example: s/he can state that 'color of car' is of high importance, and 'carbon emission', 'small size of car' and 'gasoline consumption' are of less. By doing so the customer then selects a car that best complies with all these criteria.

Compensatory strategies aim to provide the best possible decision outcomes given the decision data at hand. However, compensatory strategies require full information on how alternatives score on all criteria, and they are time consuming [11].

Noncompensatory strategies [11], on the other hand, are consistent with the concept of bounded rationality. This means that the rationale to make a decision is limited by factors such as hard constraints, time constraints and the cognitive load of the decision maker. As such, noncompensatory strategies evaluate alternatives heuristically, using only a limited number of criteria and trade-offs.

Considering a noncompensatory decision strategy for our car buying example, let us now assume that the customer lives in the city and selects between two cars: a small and a big one. Now, 'small size of car' is a hard constraint for the customer due to the limited parking space available in the city. Therefore, regardless of the criteria 'carbon emission', 'color of car', and 'gasoline consumption' s/he excludes the big car from her/his choice set.

The main characteristics of noncompensatory strategies are twofold: (1) they reduce the decision making effort, (2) they are not demanding regarding the information needed to make the decision. As such, it is a common practice for decision makers to use noncompensatory strategies in situations (time stress, hard constraints) the limitations of which affect the decision making process [15]. However, by definition decision makers do not take all criteria into account when using noncompensatory decision strategies.

Last but not least, in some cases the use of a combination of compensatory and noncompensatory decision strategies (a hybrid) is required [16], [17], [14]. For example, a decision maker starts his evaluation process by excluding alternatives that do not meet certain noncompensatory criteria, and only thereafter evaluates the remaining alternatives with a compensatory strategy.

Regarding the relationship between requirements engineering and decision making there is already a tight bond. The research community has already identified that the prioritization of requirements by using advanced techniques from decision science can improve the resulting quality of design decisions. Moreover, there are already approaches which consider software architecture as a set of design decisions [4]. In this perspective, requirements and especially the non functional ones play the role of the quality criteria which determine the selection among alternative designs and implementations. In this decision centric context it is of high importance to have approaches that would enable the architect to manage and prioritize among the given requirements. By doing so the quality of design decisions would be better [18].

III. REFINED METAMODEL

In this section we present the motivation behind the development of the refined refined metamodel and the newly introduced concepts. Furthermore, based on the evaluation of our existing approach, we have reconsidered the existing concepts and their interrelationships and multiplicites.

A. Motivation

The idea behind the enhancement of our metamodel with concepts from the requirements engineering domain is based on existing approaches from the domain of software architecture. In this domain requirements play an important role

in the selection of alternative designs. We argue that the role of requirements in enterprise architecture is even more important than software architecture. This is because of the heterogeneity of the domain of enterprise architecture. The enterprise architecture consists of completely different in nature structural elements, like business processes, hardware artifacts etc. The role of the architect is to guarantee that the enterprise architecture has a homogeneous behaviour. Therefore he should check whether the different structural elements comply with the desired behavior. However this means that a desired behavior in the business layer can be achieved by completely different in nature requirements than in the application layer or technology layer. A mechanism should enable the tracing from specific design decision in the enterprise architecture back to the requirements and vice versa.

B. Metamodel

Figure 1 presents the refined metamodel. This metamodel focuses on capturing (1) decision making strategies that were used during the architectural design process for a specific EA decision, (2) the rationale behind this specific decision strategy choice (3) the role of the functional and non functional requirements during the decision making process and (4) available alternatives that were taken into account.

Requirement: Requirements are defined as statements of need, conditions or capabilities that should be realized by a system [19], [20]. These statements can range from high level of abstraction until very detail descriptions of system functions. Moreover requirements can describe the constraints that are generated during the requirements engineering process.

The role of requirements in enterprise architecture is very essential since they must be applied in different layers / domains of the enterprise and that means that they should be interpreted by the architect in different ways depending on the domain. For example, a generic requirement for security must be realized in different ways across the different layers of the enterprise. The enterprise architect should translate this generic need into specialized requirements per domain. From the other hand an abstract requirement in a specific domain can be decomposed further in more concrete requirements in the same domain.

Our metamodel covers two different types of relationships between requirements as well as two different types of requirements. We define these types as follows:

Relationship types:

Based on our previous work [8] we have defined 2 different types of relationships between the requirement concept:

1) Decomposition:

As we can see in Figure 1 the requirement can be decomposed further until we arrive to a desired abstraction level for the domain architecture. For our approach this means that we can decompose the problem until we arrive to design decisions that lead to a concrete EA element. The decomposition relationship is in line with 'decomposes into' relationship of Kruchten's ontology [21].

2) Translation:

Translation relationship describes how requirements that belong to different EA layers / elements are translated in requirements on a different layer / artifact [8]. This relationship is critical for the domain of enterprise architecture since it provides to the enterprise architect a holistic view of the cross layer dependencies of the requirements.

Requirement types:

Requirement types can be further categorized in two types, functional and non functional requirements [22], [23]. This distinction is of high importance in enterprise architecture because the selection of specific requirements of one type can influence the other. Before we go into details on this, we describe the two different types:

a) Functional requirements:

Functional requirements specify **what** the system should do or in other words a specific behavior that a system must have [24]. It is worthwhile here to mention that even though the definition of functional requirement is new for the approach, the semantic behind this approach was partially existent. What we mean here is that the functional requirements after they have been translated or decomposed describe EA design issues that should be addressed with concrete EA design decisions. This is aligned with the semantic of the concept 'EA Issue' of the EA Anamnesis approach [8], [7]. According to the definition 'an EA issue represents the architectural design problem that enterprise architects have to address during the Enterprise transformation process.' During the refinement of the metamodel we decided to replace the term 'EA issue' with 'functional requirement' for the following reason: functional requirements, as was stated before, cover different levels of abstractions. By using the decomposition and translation relationships we can reach the abstraction level which is adequate for describing specific design issues. As a result we avoid unnecessary information redundancy and we signify explicitly that our approach can model and trace the given functional requirements. Moreover, our approach can now easily integrated with existing requirement engineering techniques.

b) Nonfunctional requirements:

Nonfunctional requirements specify the behavioral aspects of a system or in other words the quality criteria that determine **how** the system works. Furthermore other important factors like design and implementation constraints, legal requirements and project management budget and release dates are categorized as non-functional requirements [25].

Similarly with the functional requirements concept, EA Anamnesis approach was partially covering the semantic of non functional type through the 'criterion' concept. However the criterion concept was covering semantically only very concrete quality attributes. When we want to refer to more generic non functional requirements the expressivity of criterion concept was not sufficient.

From the other hand, in the case of evaluation of alternatives based on their qualities, we need a lower level of abstraction which is achieved through the decomposition / translation relationships of non functional requirements. In that case the concept 'no functional requirement' can be

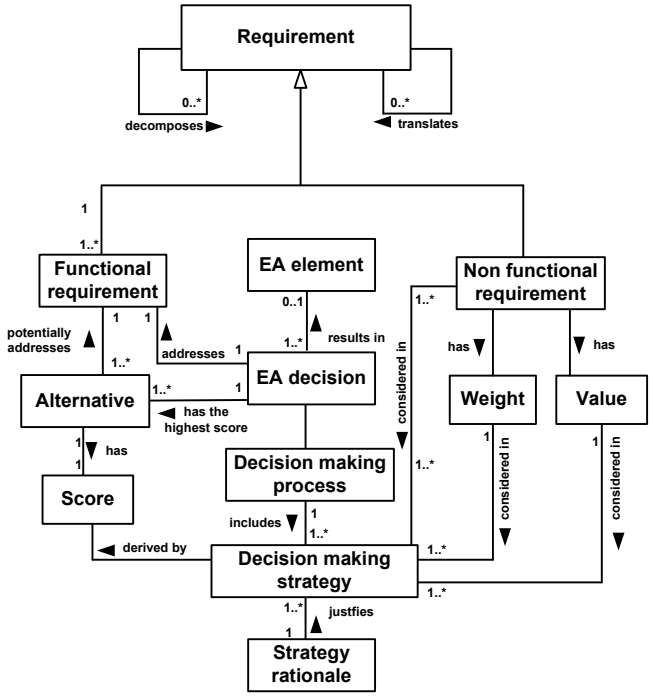

Fig. 1. Refined metamodel

used like the concept 'criterion' of our previous work [9] for evaluation in different decision making strategies. Depending on the decision strategy that was used for the evaluation process, nonfunctional requirements can be compensatory or noncompensatory. For example, if a disjunctive strategy was used, the criteria that were used for the evaluation with this strategy are disjunctive. Furthermore, the concepts **value** and **weight** of criterion are included in our metamodel. Value concept represents the value that the decision maker assigns to the non functional requirement during the evaluation process and weight concept represents the importance of the non functional requirement. Weight concept is used in WADD strategies. This gives the ability to the architect to trace back the decision making process and analyze the value as well as the importance of each non functional requirement during the evaluation process. By capturing the type, value and weight of non functional requirements, stakeholders that analyze in depth the architecture, can understand which criteria had a determinant role in the selection process and on which strategy they were based.

Last but not least the design of our metamodel can express relationships between different types of requirements, for example decomposition from a functional requirement to a nonfunctional requirement or vice versa [24]. For example the non functional requirement 'security' may be translated to the functional requirement 'encryption' in the application layer.

Decision making process: A decision making process provides the summarization of the decision making for a specific design decision. As we explain below one or more decision making strategies can be used for the making of a single decision. In our metamodel the decision making process is automatically derived by the set of the the decision making strategies and it is linked to the concept of EA decision.

Decision-Making Strategy: This concept captures the decision making strategy used by the enterprise architect to (1) evaluate the alternatives, and make the actual EA decision. As we mentioned in Section II-A, decision strategies are characterized as compensatory, noncompensatory, or as a hybrid of these two. In our metamodel, we specify this as follows:

- **Compensatory strategy**
  - Weighted additive (WADD): In WADD strategies the criteria that evaluate the alternatives have different weights. The score of each alternative is computed by multiplying each criterion by its weight and then by taking the sum of these values. The alternative with the highest score is chosen by the decision maker [14].
  - Equal weight: The score of each alternative is calculated by the same way as WADD strategies. The difference is that criteria have the same weight [14].
- **Noncompensatory strategy**
  - Conjunctive: In conjunctive strategies, alternatives that fail to comply with a minimum threshold level of one or more criteria, are immediately excluded from the decision maker's choice set [14].
  - Disjunctive: In this strategy alternatives are evaluated based on the maximum threshold level of one or more criteria. Those which fail to meet the maximum level, are excluded from the choice set [14].

We should also mention that there is no restriction in the use of additional decision strategies. We include a set of common decision strategies, but we also denote in the strategy viewpoint metamodel that more decision strategies can be supported.

Strategy rationale: In the context of a decision making process, the architect not only has to choose amongst some alternatives (actual decision making process), but has also to select the decision strategy that satisfies his current evaluation needs. Typical reasons for the adoption of different strategies by the architect are constraints that come from different domains of the enterprise. The capturing of this information justifies the decision strategy that was selected for the evaluation process. This is what is referred as metadecision making, decision making about the decision process itself [26]. As stated in the metamodel one strategy rationale can justify one ore more decision making strategies.

EA decision: The concept of EA decision represents the design alternative which has the maximum score (utility) after the execution of the decision making process. Since the information for the alternatives already exist the concept 'EA decision' is automatically derived from the alternative with the highest score.

Alternative: The concept of alternative represents the available choices that are under evaluation by using a specific decision making strategy.

IV. ILLUSTRATIVE EXAMPLE

We now show how the EA Anamnesis approach captures decision making processes as well as the influence of requirements on them. The refined metamodel enhances the rationalization information that EA Anamnesis provides. For illustration purposes, we use an insurance company case study presented in our previous paper [27].

A. ArchiSurance: moving to an intermediary sales model

ArchiSurance is an insurance company that sells insurance products using a direct-to-customer sales model. The company used this disintermediation scheme to reduce its operations and product costs.

Although, disintermediation reduces operational costs, the use of intermediaries in insurance sector is very important because they provide accurate risk customer profiles [28]. ArchiSurance management decides to adopt this practice and to change its selling model to intermediary sales. The role of the Insurance broker is added to the business operation of the company.

B. Capturing the influence of requirements on decision process for ArchiSurance

In our scenario, an external architect called *John* is hired by ArchiSurance to change the Enterprise Architecture and analyze the impacts that the intermediary sales of insurance has on the company.

John uses the EA modeling language 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 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'.

However, John (by using ArchiMate) can not capture the rationale behind this model. For example, he captures the change for the different application system that supports the new business process, but he is not able to justify why and how he selected this specific system among other alternatives.

To capture design rationales behind the ArchiMate model, John relies on the EA Anamnesis approach (our previous work, see [27], [7]). Table I shows an example application of the EA Anamnesis approach (EA decision 13). As it can be observed, decision facets such as the decision rationale (why the decision was taken), criteria (factors, such as cost), observed impact (ex-post) are captured. For further details, see [27].

However, from a requirements engineering point of view the traceability of the functional and nonfunctional requirements is not explicitly provided by the EA Anamnesis approach. Moreover we are not able to identify the role of these requirements on the decision making process of John. Therefore, we now replaying the decision making process which leads to the creation of the decision 13 displayed in Table I in order to show the role of the functional and non functional requirements to this decision.

Capturing the influence of requirements on decision process for ArchiSurance:

For the purposes of this paper, we focus on capturing and analyzing the influence of requirements on the decision making

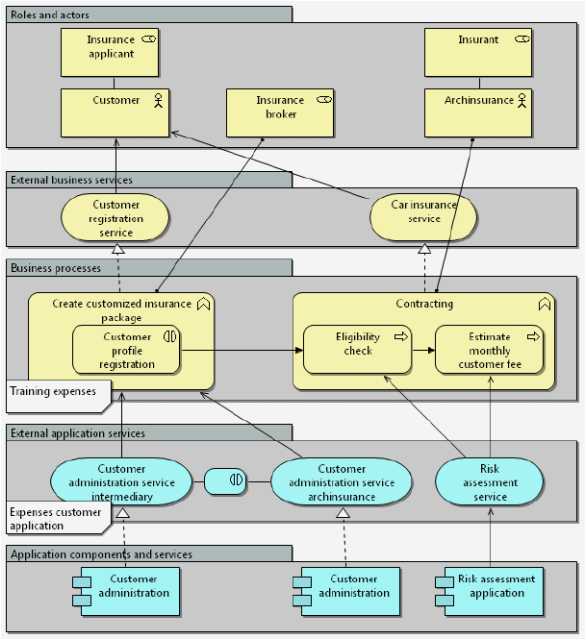

Fig. 2. ArchiSurance intermediary EA model

TABLE I. EA DECISION 13 DETAILS

Title:	Acquisition of COTS application B
EA issue:	Current version of customer administration application is not capable to support maintenance and customers administration of intermediaries application service
Decision Maker:	John
Layer:	Application
Intra-Layer dependent Decisions:	EA decision 10
Inter-Layer dependent Decisions:	None
Alternatives:	COTS application A COTS application C Upgrade existing application (inhouse)
Rationale:	Scalability: Application is ready to support new application services
Criteria:	Customized reports capability, interoperability, scalability, cost
Observed Impact:	Interoperability issues. Application COTS B can not communicate with existing applications of some insurance brokers.

process. Therefore, we assume that John is a single decision maker, who is capable to identify the above concepts and he has full information to evaluate them. We thus abstract away from the identification of specific alternatives, requirements and their respective scores.

Figure 3 visualizes the role of the requirements on the decision making process. We can first observe how this architecture scenario is analyzed as a set of functional and nonfunctional requirements. The need for a new business process that would support the insurance broker role is expressed by the functional requirement 'Establish business process intermediary'. Subsequently, this functional requirement creates another requirement 'Find an appropriate application to interface with the intermediary' in the application layer of the enterprise. The new business process should be supported by an appropriate application system. As we explained in the metamodel section, EA Anamnesis captures the relationship of requirements that they belong in different architecture layers with a translation relationship. Furthermore in the context of the specific application system John captured the functional requirement 'Find user interface for the application'. Since this requirement describes a subproblem for the same architectural element we use a decomposition relationship to connect it with the 'Find an appropriate application to interface with the intermediary' requirement.

Let us now zoom into the decision making process for design decision 13. John made this decision in order to address the functional requirement 'Find an appropriate application to interface with the intermediary'. John now uses the EA Anamnesis approach to capture the role of the non functional requirements on the decision making process. Initially he captures how more generic non functional requirements which are defined during the initial phases of the enterprise architecture transformation are decomposed into more detailed non functional ones which can be used for the evaluation of the alternatives. This is expressed by the decomposition relationship between non functional requirements. For this specific case the generic requirement 'IT Systems Adhere to Open Standards' is decomposed in the non functional requirement 'interoperability'. As a next step he uses EA Anamnesis, to capture the non functional requirements which were used for the evaluation of the alternative solutions (the requirements for application selection are grounded in [29]).

For our illustrative example, John considers that the most important non functional requirements are 'customized reports capability', 'interoperability' and 'scalability'. Based on these requirements he identifies four alternatives to choose from: three alternative Commercial Off-The-Shelf (COTS) applications and one to upgrade the existing application in house.

Let us also assume that John receives a budget constraint of 10000$ for the acquisition of new IT systems. As we described in the metamodel section constraints are also expressed as non function requirements

John is now faced with a hybrid decision strategy: on the one hand, he wants to carefully evaluate the four alternatives on the non functional requirements 'customized reports capability', 'interoperability' and 'scalability' (via a compensatory strategy), but on the other hand John has to account for the hard constraint of 'budget limitation' (via a non-compensatory strategy).

At this time John uses the decision making strategy part of the approach to capture and justify his strategy selection, as well as the alternatives and non functional requirements of his decision problem. For the noncompensatory part, John wants to discard all alternatives that fail to meet the budget constraint. Because of this hard constraint, he chooses a disjunctive noncompensatory strategy (for an explanation, see Section III-B) to exclude from his choice set alternatives that exceed the maximum budget cost.

Table II summarizes the score of each alternative. COTS application C is eliminated from the choice set because it failed to meet the maximum cost requirement.

As we mentioned before, disjunctive noncompensatory strategies evaluate alternatives using a maximum threshold level on one or more criteria. In this example the disjunctive criterion is 'cost'. The alternatives 'COTS A','COTS B' and 'Upgrade application' comply with this criterion (Table II) and will be evaluated further in the next step of the decision making process. 'COTS C' cost exceeds the maximum budget limit and it is eliminated from the choice set. For noncompensatory strategies, alternatives either comply or not to some criteria and their score are Boolean data types. The scores of the alternatives are also captured by our metamodel.

For the compensatory part, John evaluates the three remaining alternatives based on the values and the weight of each of the criteria.

Scalability is the most important factor because, according to John, this application should be able to support changes in the business processes of ArchiSurance. This is important in order to support the addition of extra intermediaries. Customized reports capability and interoperability are also important, not as important as scalability.

TABLE II. EA DECISION 13 NONCOMPENSATORY DISJUNCTIVE STRATEGY

Alternatives	cost	score
COTS A	9000$	1
COTS B	8000$	1
COTS C	12000$	0
Upgrade app	5000$	1

TABLE III. EA DECISION 13 COMPENSATORY WEIGHTED ADDITIVE STRATEGY

Alternatives	custom reports	interoperability	scalability	score
COTS A	7x7	7x5	7x10	154
COTS B	8x7	3x5	9x10	161
Upgrade app	5x7	5x5	4x10	100

Given the fact that criteria that evaluate alternatives have different weights, John selects the use of a weighted additive compensatory strategy. At this moment John captures again his decision strategy as well as the weights and the values of the compensatory criteria. The score of each alternative is calculated by multiplying the value of each non functional requirement by its weight, and then by taking the sum of these values. Here, the weights range from 1 - not important - to 10 - important. The alternative with the highest score is chosen by the decision maker.

Table III shows us: (1) the criteria. 'Scalability', the most important non functional requirement for John, has a weight of 10, while 'Custom reports' and 'interoperability' have weights of 7 and 5 respectively, (2) the score on a particular non functional requirement for each alternative. For example: the alternative 'COTS B' scores 9 on 'scalability', whereas 'Upgrade app' scores 4. (3) the total score of each alternative. For example: 'COTS B' receives the highest score and as such, is selected by John.

Reflecting upon the captured decision making process and requirements. So far, we have illustrated the capturing of design rationales by John. Let us now illustrate how the information can be useful by the new enterprise architect, Bob.

Bob's predecessor, John, captured the decision making process and the role of requirements with the EA Anamnesis decision strategy viewpoint. Bob can now analyze (1) the used strategy, (2) why this strategy was selected, and (3) the role and importance of requirements for this evaluation process.

Figure 3 shows the decision making process for EA decision 13 based on the refined metamodel. From the decision making steps 'Conjunctive strategy because of budget constraint' and 'WADD strategy different NFR weights' , Bob understands that a hybrid decision making strategy model was used. More specifically, he realizes that because of a budget constraint one of the alternatives was discarded with the use of a disjunctive noncompensatory strategy. Furthermore, Bob observes that a compensatory weighted additive strategy was used to evaluate the remaining alternatives. He realizes that his predecessor used this strategy, because the non functional requirements 'customized reports capability', 'interoperability' and 'scalability' did not have the same importance for the selection of an appropriate application that would support the new business process of ArchiSurance. He can also see the weight of each non functional requirements, as well as the final score of each of the alternatives.

This reconstruction of the decision making process makes transparent *how* an EA design decision has been made. Amongst other, this transparency allows an architect to compare the outcome of an EA decision with the non functional requirements that influenced this decision. As a result, s/he can learn which factors in the decision making process had a positive/negative impact to the EA design and follow/avoid these decision making practices for future decisions.

After a period of time, COTS application B does not have a sufficient performance due to interoperability issues. Bob, is asked by management to explain the choice for COTS application B. He can reconstruct and examine the decision making process using Tables II and III. First, he observes that COTS application C was eliminated because of budget issues. Second, from the weight assigned to the different non functional in Table III, he observes that scalability was an important criterion for his predecessor to select COTS application B, but not interoperability. By examining the captured non functional requirements and their weights as well as the observed impact of the EA Decision (Table I) Bob can learn that interoperability is an important non functional requirement for Archisururance enterprise architecture and should be weighted and compared against other non functional requirements more carefully. For example: in a future decision making process, Bob can provide a weight of 7 or 8, instead of 5, to the non functional requirement interoperability to better weight it against other criteria such as scalability.

V. RELATED WORK

Concerning **argumentation-based** approaches, The Decision Representation Language (DRL) [30] and Issue Based Information System (IBIS) [31] are two well known approaches for capturing design rationale. Both DRL and IBIS are inspired by Toulmin's analysis of argumentation [32] and argumentation maps. For DRL and IBIS, key rationalization concepts are the issue, the arguments and the resolution of design argumentation. Here, for example, resolutions are similar to Toulmin's conclusion of an argument.

However, as pointed out by Shipman III and McCall [33], argumentation-based approaches are not suitable for capturing communicating design rationales in practice. Mainly this is because argumentation-based approaches require extensive documentation [33]. Furthermore, argumentation-based approaches lack formality, and thus are not amenable to the computer-based support that we aim for with EA Anamnesis. Also argumentation-based approaches do not make an explicit relation to the design artifact under consideration, while for us it is important to focus our rationalization on particular EA artifacts (such as elements of architectural languages).

Moreover, since its second version the **Archimate EA modeling language** has a motivation extension and an implementation and migration extension. The motivation extension is used to model the reasons behind architectural changes, but lacks concepts common to existing rationalization approaches. For example, the motivational extension does not capture design alternatives, the used decision making strategy, or unanticipated consequences of decisions. Furthermore the Implementation and Migration Extension deals with the project management and the planning of enterprise architecture changes and as such, is not well suited for architectural rationalization.

Finally, there exist **design rationale approaches for Software Architecture** [4][5][34][35][36]. These approaches are template based or model based. Akin to argumentation-based rationalization approaches, template based approaches [34][36]

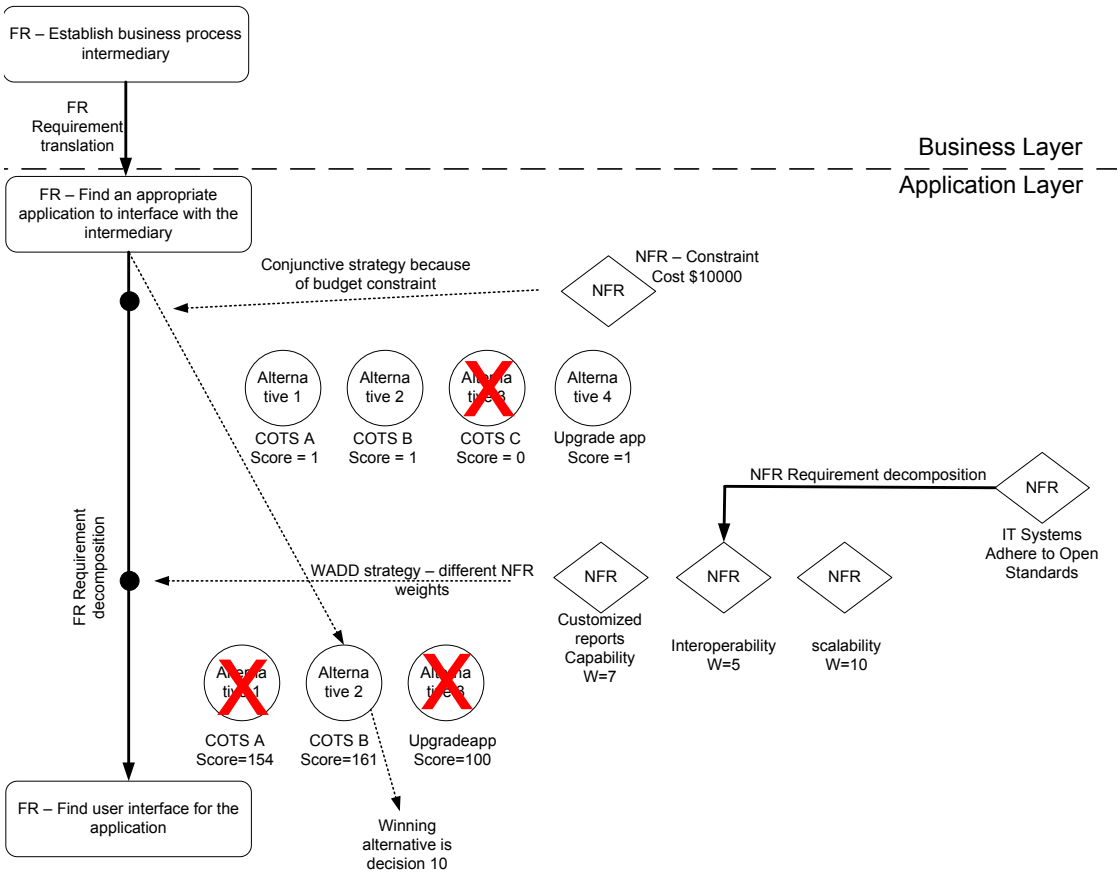

Fig. 3. Decision making process and requirements traceability of design decision 13

describe *in textual format* elements of Architectural Decisions such as Rationale, Issue, Implications and etc. Differently, model based approaches [4][5][35] provide a *formal metamodel* of decision rationalization concepts, thus enabling computer-processable rationalization.

However rationalization approaches from software architecture focus on software issues (such as code documentation). Yet these issues are different from those in Enterprise Architecture [37]. For one, as part of their responsibilities enterprise architects concern themselves with the business-to-IT-stack [38]. Here, for example, they analyze impacts of changes in IT infrastructure to business processes and vice versa. Thus enterprise architects deal with different, *cross-organizational*, issues compared to software architects, who deal mostly with software concerns.

VI. CONCLUSION

In this paper we introduced a metamodel for capturing the the influence of functional and non functional requirement on the decision making processes in enterprise architecture. Furthermore, we argued about the role of the requirements on the decision making process and why it is useful to capture such information (1) by argumentation, (2) by an illustrative case study.

For future research, we aim to continue our evaluation activities by means of case studies. In the next iteration of the evaluation we will focus on indenting to what extend the requirements influence the decision making process of practitioners and if the EA Anamnesis approach helps them to capture and analyze this information.

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 in the decision making strategy viewpoint. For example we capture the strategy rationale for selecting a decision making strategy, but it remains to be seen if the effort for capturing this, outweighs the received benefits.

ACKNOWLEDGMENTS.

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

REFERENCES

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

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

[3] R. Winter and R. Fischer, "Essential layers, artifacts, and dependencies of enterprise architecture," *Journal of Enterprise Architecture–May*, pp. 1–12, 2007.

[4] 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.

[5] 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

[6] G. Plataniotis, S. de Kinderen, D. van der Linden, D. Greefhorst, and H. A. Proper, "An empirical evaluation of design decision concepts in enterprise architecture," in *Proceedings of the 6th IFIP WG 8.1 working conference on the Practice of Enterprise Modeling (PoEM 2013)*, 2013.

[7] G. Plataniotis, S. de Kinderen, and H. A. Proper, "Ea anamnesis: An approach for decision making analysis in enterprise architecture," *International Journal of Information System Modeling and Design (IJISMD)*, to appear.

[8] G. Plataniotis, S. d. Kinderen, and H. A. Proper, "Relating decisions in enterprise architecture using decision design graphs," in *Proceedings of the 17th IEEE International Enterprise Distributed Object Computing Conference (EDOC)*, 2013.

[9] G. Plataniotis, S. de Kinderen, and H. A. Proper, "Capturing decision making strategies in enterprise architecture – a viewpoint," in *Enterprise, Business-Process and Information Systems Modeling*, ser. Lecture Notes in Business Information Processing. Springer Berlin Heidelberg, 2013, vol. 147, pp. 339–353. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-38484-4_24

[10] D. Greefhorst and E. Proper, *Architecture Principles: The Cornerstones of Enterprise Architecture*. Springer Science & Business Media, 2011, vol. 4.

[11] H. Einhorn, "The use of nonlinear, noncompensatory models in decision making." *Psychological bulletin*, vol. 73, no. 3, pp. 221–230, 1970.

[12] 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.

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

[14] L. Rothrock and J. Yin, "Integrating compensatory and noncompensatory decision-making strategies in dynamic task environments," *Decision Modeling and Behavior in Complex and Uncertain Environments*, pp. 125–141, 2008.

[15] J. Payne, J. Bettman, and E. Johnson, *The adaptive decision maker*. Cambridge University Press, 1993.

[16] T. Elrod, R. Johnson, and J. White, "A new integrated model of noncompensatory and compensatory decision strategies," *Organizational Behavior and Human Decision Processes*, vol. 95, no. 1, pp. 1–19, 2004.

[17] I. Jeffreys, "The use of compensatory and non-compensatory multi-criteria analysis for small-scale forestry," *Small-scale Forestry*, vol. 3, no. 1, pp. 99–117, 2004.

[18] A. Aurum and C. Wohlin, "The fundamental nature of requirements engineering activities as a decision-making process," *Information and Software Technology*, vol. 45, no. 14, pp. 945–954, 2003.

[19] A. P. Sage and W. B. Rouse, *Handbook of systems engineering and management*. John Wiley & Sons, 2009.

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

[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] P. Loucopoulos and V. Karakostas, *System Requirements Engineering*. New York, NY, USA: McGraw-Hill, Inc., 1995.

[23] E. Kavakli and P. Loucopoulos, "Goal modeling in requirements engineering: Analysis and critique," *Information Modeling Methods and Methodologies: Advanced Topics in Database Research: Advanced Topics in Database Research*, p. 102, 2004.

[24] S. Robertson and J. Robertson, *Mastering the requirements process: Getting requirements right*. Addison-wesley, 2012.

[25] L. Chung, B. Nixon, E. Yu, and J. Mylopoulos, "Non-functional requirements," *Software Engineering*, 2000.

[26] H. Mintzberg, D. Raisinghani, and A. Theoret, "The structure of unstructured decision processes," *Administrative science quarterly*, pp. 246–275, 1976.

[27] 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

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

[29] A. Jadhav and R. Sonar, "Evaluating and selecting software packages: A review," *Information and software technology*, vol. 51, no. 3, pp. 555–563, 2009.

[30] 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.

[31] W. Kunz and H. W. Rittel, *Issues as elements of information systems*. Institute of Urban and Regional Development, University of California Berkeley, California, 1970, vol. 131.

[32] S. E. Toulmin, *The uses of argument*. Cambridge University Press, 2003.

[33] F. M. Shipman and R. J. McCall, "Integrating different perspectives on design rationale: Supporting the emergence of design rationale from design communication," *Artificial Intelligence for Engineering Design, Analysis and Manufacturing: AI EDAM.*, vol. 141, p. 141, 1997.

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

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

[36] 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.

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

[38] S. Aier and R. Winter, "Virtual decoupling for it/business alignment - conceptual foundations, architecture design and implementation example," *Business & Information Systems Engineering*, vol. 1, no. 2, pp. 150–163, 2009. [Online]. Available: http://dx.doi.org/10.1007/s12599-008-0010-7