EA Anamnesis: Towards an approach for Enterprise Architecture rationalization

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
e.proper@acm.org

ABSTRACT

ArchiMate [4] is a Domain Specific Language (DSL) to model an enterprise from a holistic perspective, showing not only the IT infrastructure of an organization, but also how this IT infrastructure supports business processes and contributes to the realization of products and (commercial) services.

Yet, ArchiMate lacks the capability to capture the design decisions behind the models. Capturing such decisions is important to improve the design teaching and communication after the design process. This is what we refer to as EA Anamnesis. People who can benefit from EA Anamnesis are e.g. persons that are foreign to a given architecture, such as external Enterprise architects.

In this paper, we introduce an approach to capture design decisions. For the moment we target our approach primarily on capturing design decisions in the context of ArchiMate models. Specifically, we (1) introduce a metamodel for capturing architectural design decisions. This metamodel is grounded in DSLs for capturing rationales in software engineering. (2) Finally, we provide a fictitious use case scenario for the insurance industry to illustrate the use of our approach.

Categories and Subject Descriptors

K.6.1 [**Management of computing and information systems**]: Project and People Management—*Systems analysis and design*; D.2.2 [**Design Tools and Techniques**]: Decision tables

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

General Terms

Theory, Languages, Design, Documentation

Keywords

Enterprise Architecture, Domain Specific Language, Design Rationale, Decision Capturing

1. INTRODUCTION

ArchiMate is a Domain Specific Language from the Open Group for the modeling of Enterprise Architectures [11][5]. By modeling enterprise architectures, one captures an organization *holistically*, amongst others linking 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 process. Such a holistic perspective on an enterprise helps to clarify the business advantages of IT [11], analyze cost structures and more [8].

While ArchiMate allows for modeling an enterprise holistically, the design decisions behind the resulting models are often left implicit. Although we should be careful with the analogy, experience from the field of software architecture shows that leaving implicit design rationales leads to "Architectural Knowledge vaporization" (cf. [6]). 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, such lack of transparency regarding design decisions can cause design integrity issues when architects want to maintain or change the current design [15]. This means that due to a lacking insight in the rationale, new designs are constructed in an ad/hoc manner, without taking into consideration constraints implied by past design decisions. Also, according to a survey for software architecture design rationale [14], a large majority of architects (85,1%) admitted the importance of design rationalization in order to justify designs. Another interesting finding of this survey was that architects declared that after some time they frequently forget their own decisions. Moreover, anecdotal evidence from six exploratory interviews that we had with senior enterprise architects, suggests that Architects are often external consultants. This situation increases the architectural knowledge gap of the Enterprise Architecture because the successor architect tries to understand and analyze the architecture by searching through architectural designs and unstructured information of requirements documentation.

Although the recently released ArchiMate 2.0 version [4] has a motivational layer, it has not been designed specifically for capturing design decisions in models in the broad sense. As such, ArchiMate 2.0 does not capture concepts standard to design rationale languages such as DRL [9]. Examples of such decision concepts include alternatives, selection criteria, policies and many more.

In this paper, we introduce a DSL for the domain of capturing enterprise architectural design decisions modeled in ArchiMate. We call this DSL EA Anamnesis. Anamnesis ($\alpha\nu\acute{\alpha}\mu\nu\eta\sigma\iota\varsigma$), an ancient Greek word, is a term used in medicine, philosophy and other sciences and denotes reminiscence and repair of forgetfulness. The specific contribution of this paper is: (1) to introduce a formal metamodel and representation thereof to formally capture design decisions made by enterprise architects and (2) to show the use of capturing architectural design decisions.

This paper is structured as follows. Section 2 introduces the EA anamnesis approach and metamodel, while section 3 illustrates the use of our approach with an insurance example. Section 4 presents related work. Finally section 5 concludes.

2. THE EA ANAMNESIS APROACH

In this section we introduce the idea of EA anamnesis (Section 2.1), a formal approach for capturing and representing Architectural design decisions. The EA anamnesis metamodel, central to our formal approach, will be discussed in Section 2.2.

2.1 EA Anamnesis

We introduce an approach for capturing Enterprise Architecture Decisions. With this approach we contribute to the domain of Enterprise Architecture by assisting architects to better understand the existing Architecture of the Enterprise, especially in terms of the rationales behind the architecture. Our approach is based on Decision Representation Language (DRL) [9] and a formalism for organizing and representing decisions using Design Decisions Trees [12]. DRL is a well-established language for decision modeling that has also used to model Software Architecture decisions [10][16].

More particular we propose a specialized template that captures an EA decision that is taken, in terms of attributes that are based on DRL and design decision trees, combined with attributes that are specific to ArchiMate. Examples of such attributes include the actual decision, the issue to which the decision responds, the decision rationale and the impact that the decision has on other layers of the enterprise. Also, we provide the basis for the creation of an EA DDT (Decision Dependency Tree), a representation scheme that is used to make explicit the impacts that a decision has across an organization.

2.2 EA Decisions metamodel

We now discuss each of the concepts from the EA decisions metamodel, depicted in Figure 1. This metamodel is instantiated in decision tables and decision trees, which are discussed in further detail in Section 3.

Title: A small but informative description of the EA decision that was taken. The information of this field can also be used when we represent the whole set of organizational decision with Design Decision Trees.

EA issue: This field addresses the issue that the Enterprise Architect had to solve with this decision. It is a short description of the issue and not an argumentation for the decision.

Decision Maker: The stakeholder that was responsible for this specific decision.

Layer: In line with the ArchiMate language [4] 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. To allow for a holistic expression of an enterprise, layers are (1) self-contained, meaning that each layer contains its own structural model elements and (2) integrated, meaning that layers and their elements are interrelated to each other with special relationship types.

Rationale: The reason that leads the architect to choose the specific decision among the alternatives. According to Kruchten [7] a rationale answers the "why" question for each decision. The information of this field should provide some added value to the overall decision of the information.

Alternatives: This field describes the alternative choices that were considered for the same EA issue. An architect can use alternatives to describe the alternative options for addressing the EA issue.

Criteria: This field exposes specific criteria that influenced the decision maker during the selection of the specific decision among the alternatives. For example, a selection criteria for a decision may include development time when weighing the in-house development of an application against a common-of-the-shelf application.

Policy: This field declares possible Organizational policies that should be applied during the Decision Selection process. Examples of such policies are cost reduction, confidentiality, availability etc. For the purposes of this paper, this field helps Enterprise Architects to better understand if a specific decision was taken not only based on some criteria and rationale, that were related to a particular domain, but also based on organizational policy.

Observed Impact: With this concept, we record observed impacts of the EA decision. In line with the ex-post nature of the EA Anamnesis approach presented in this pa-

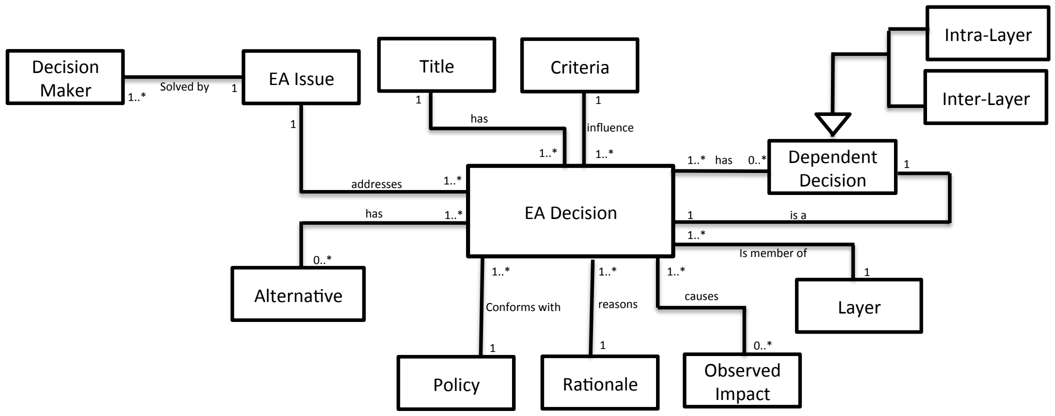


Figure 1: EA Decisions Metamodel

per, this means the recording of *actual* impacts, not anticipated ones. These could be positive/negative or predictable/unpredictable circumstances that occur after the application of the current decision. For example, the decision for a new application service causes network unavailability because of increased network traffic. Enterprise Architects can use information of this field in order to avoid decisions with negative impacts in the future, to be aware of possible negative impacts their decision may have.

Dependent decision: Here we represent decisions that are affected by the EA decision. We distinguish between two types of dependent decisions:

- **Intra-Layer dependent Decision:** This field contains decisions of the same Layer that that have a dependency relationship with the specific decision. For example, in Technology layer, a decision "install additional servers" leads also to the decision "extension of network infrastructure".
- **Inter-Layer dependent Decision:** This field contains decisions from different Layers that have a dependency relationship with this decision. Importantly, using the concept of an inter-layer dependent decision we can construct the Decision Tree structure that represents how decisions from upper layers affect decisions from lower layers. For example, a decision for a new business service "let customer apply for insurance online", taken on the business layer, triggers other decisions on other layers of the enterprise, such as "install additional servers" on the technology layer.

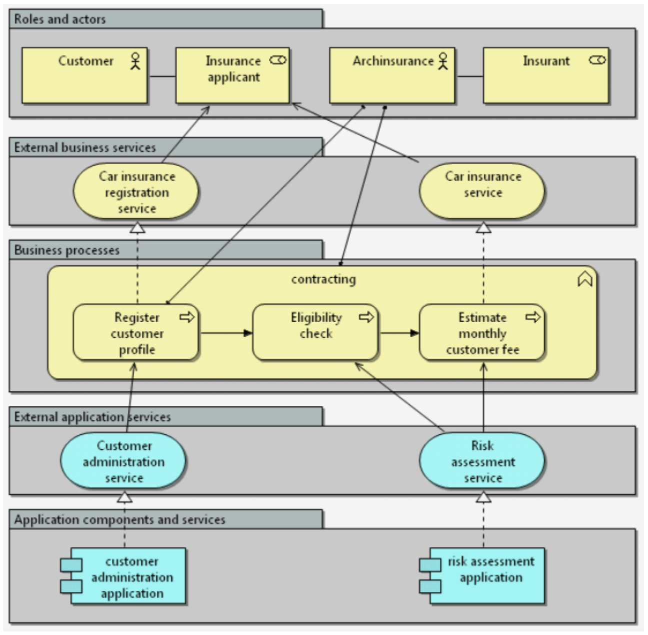
3. ILLUSTRATIVE EXAMPLE

We now show how the EA decisions metamodel can be used to express architectural design rationales and how subsequently these design rationales can be used to trace back design decisions. For illustration purposes, we use an insurance company case study presented in a paper for model integration [3]. To demonstrate how our approach complements Enterprise Architecture Description Languages we use partial (selected layers) Enterprise Architecture views from ArchiMate (Figures 2, 3). We aim to illustrate that the proposed approach can assist Enterprise Architects in understanding the existing (as-is) architecture by capturing details about Enterprise Architecture Decisions.

3.1 ArchiSurance: moving to an intermediary sales model

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

Figure 2 presents the partial (Business and Application layers) ArchiMate model for ArchiSurance direct-to-customer sales model. Two business services support the sales model of ArchiSurance, "car insurance registration service" and "car



Roles and actors
Customer
Insurance applicant
Archinsurance
Insurant
External business services
Car insurance registration service
Car insurance service
Business processes
contracting
Register customer profile
Eligibility check
Estimate monthly customer fee
External application services
Customer administration service
Risk assessment service
Application components and services
customer administration application
risk assessment application

Figure 2: ArchiSurance direct-to-customer EA model

insurance service". ArchiMate helps us understand the realization dependencies between different elements. For example, in Figure 2 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, the use of intermediaries in insurance sector is very important because they provide accurate risk customer profiles [2]. ArchiSurance management decided to adopt this practice and to change its selling model to intermediary sales. The role of the Insurance broker was added to the business operation of the company.

3.2 Capturing the intermediary sales model in EA anamnesis

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

Scenario: 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 3.

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't capture the rationale behind this model. For example, he captured the change for the different application architecture that supports the new business process but he wasn't able to capture the justification for his decision. To capture design rationales behind the ArchiMate model, John relies on the EA anamnesis approach.

For this simplified scenario 13 Architectural Decisions were taken. These decisions and their relationships are represented in the DDT structure (Figure 4) that we explain later. Table 1 shows an example of a single decision, captured in a decision table by John: "upgrade customer administration application".

Let us assume that a newly hired Enterprise Architect, Bob, wants to understand the Enterprise Architecture of the organisation by using EA Anamnesis. Bob is interested in inspecting the decision in Table 1 "upgrade of customer administration application". Tables provide us structured information about what decision was taken (Title), for what (EA issue) and by whom (Decision Maker). Very important fields that justify the reasons for the particular deci-

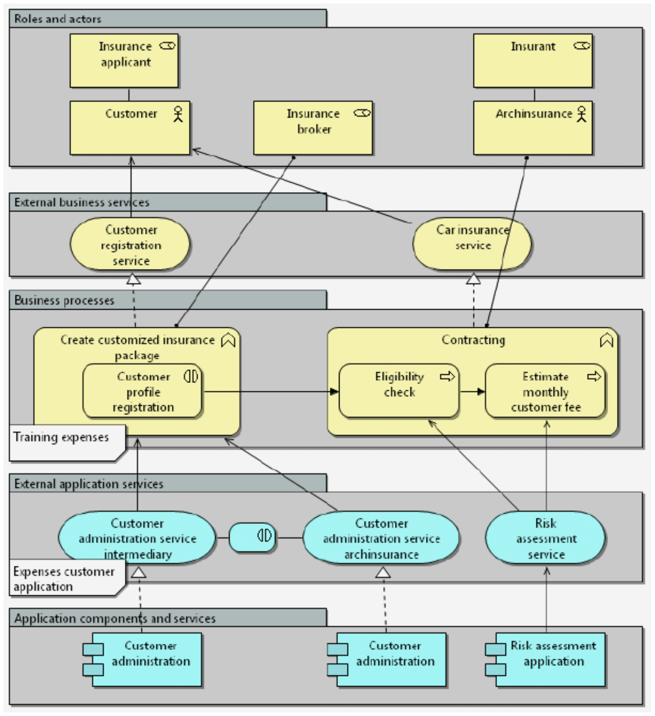

Roles and actors
Insurance applicant
Insurant
Customer
Insurance broker
Archinsurance
External business services
Customer registration service
Car insurance service
Business processes
Create customized insurance package
Contracting
Customer profile registration
Eligibility check
Estimate monthly customer fee
Training expenses
External application services
Customer administration service intermediary
Customer administration service archinsurance
Risk assessment service
Expenses customer application
Application components and services
Customer administration
Customer administration
Risk assessment application

Figure 3: ArchiSurance intermediary EA model

Table 1: EA Decision 13 details

Title:	Upgrade of customer administration application
EA issue:	Current version of customer administration application isn't 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:	Acquire Common of the shelf application
Rationale:	With the upgrade we maintained the existing Application GUI for responsible users of customer registration department. Users should only be trained to use the additional parts, the upgraded application provides, regarding customer information of intermediaries
Criteria:	Reduced Risk, Downtime
Policy:	Cost reduction
Observed Impact:	Business Layer: Increased adaptability to the new business process model because people from customer registration department just learned to work with the new information workflow model without having to use a different application

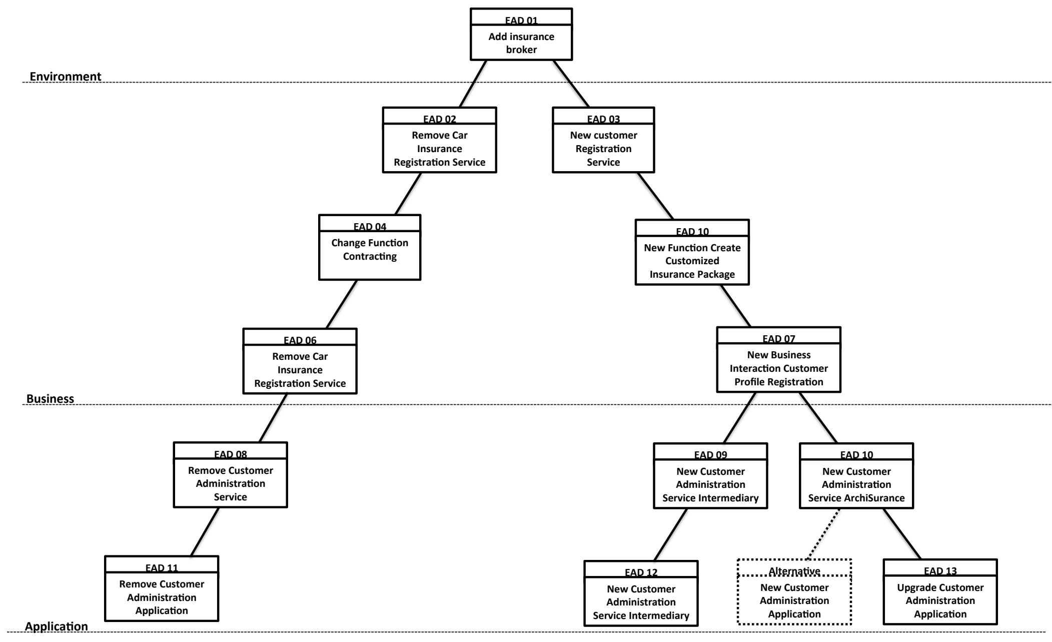

EAD 01
Add insurance broker
Environment
EAD 02
Remove Car Insurance Registration Service
EAD 03
New customer Registration Service
EAD 04
Change Function Contracting
EAD 10
New Function Create Customized Insurance Package
EAD 06
Remove Car Insurance Registration Service
EAD 07
New Business Interaction Customer Profile Registration
Business
EAD 08
Remove Customer Administration Service
EAD 09
New Customer Administration Service Intermediary
EAD 10
New Customer Administration Service ArchiSurance
EAD 11
Remove Customer Administration Application
EAD 12
New Customer Administration Service Intermediary
Alternative
New Customer Administration Application
EAD 13
Upgrade Customer Administration Application
Application

Figure 4: EA Decisions Dependency Tree structure

sion are Policy, Criteria and Rationale. The new architect has now at his disposal not only implementation details but also the reasons for this specific decision that triggered the architectural change. For example organizational policy of ArchiSurance at the time that transformation was initiated was to enforce reduction costs. Next architect can understand in depth reasons that influenced his predecessors to take a specific decision.

For EAD13 the final decision was to upgrade the administration application. Bob can determine that the alternative choice was the acquisition of a completely new application. This decision was taken because the existing application was not able to support the new application service that is described in EA issue field. Bob is also able to understand the reason that led to this decision. Reduced Risk and downtime (Criteria considered) of application upgrade, cost reduction policy (Policy) of organization and minimal training needs (Rationale) were the main factors.

Next, let us assume that Bob is interested in reviewing the impacts that the individual decision has on related decisions. He can easily understand, by examining Observed Impact field, that EAD13 had a positive impact in Car insurance Business Layer process (Business Layer) because users were already familiarized with the application Interface.

By using EA decision dependency relationships from our metamodel, decisions can be organized and represented using DDT structures. To construct the DDT structure, we start by defining the "root" node of the tree. Following a top-down approach, the "root" node is always the highest layer decision. We define the important details of this node and based on dependency information we construct the "children" nodes. "Children" nodes denote dependent decisions that realize the requirements that "parent" nodes define. Similar to [12], these triggering relationships create the decisions tree structure which represent the Architectural knowledge in a hierarchical way. This is an iterative procedure that continues until all required dependent decisions are derived. This structure provides decision traceability support. The architect can begin examining lower layers decisions and by exploring the tree, trace back to strategic goals and business requirements.

In our scenario, Bob is able to trace DDT and discover the relations with other decisions. As we can see from DDT, EAD13 was caused by EAD10 (New customer Administration Service ArchiSurance). EAD13 satisfies the requirements that EAD10 created. This type of relationship is an Intra-Layer dependent decision because both related decisions belong to the same Architecture Layer. This dependency is also denoted in the appropriate field of the template. As stated, Impact information may also provide information related with other layers. In our example EAD13 had a positive impact in new business process of ArchiSurance because users were already familiarized with the Application interface.

4. RELATED WORK

A lot of work [6][15][16][7][13] has been done in the area of Design rationale and capture of Architectural Decisions. Nevertheless, this work is focused on Software Architecture and problems that software architects have to cope with when dealing with software projects (like software design patterns etc.). Software architecture is only a subset of Enterprise Architecture [1].

There are different approaches to capture Software Architectural Design rationale. Most of them are template based or model based. Textual Template based approaches [16][13], describe in textual form important details of Architectural

Decisions like Rationale, Issue, Implications and etc. Although template based approaches provide useful information about decisions, this information is unstructured and can't be used efficiently.

Model based approaches [6][15][7] provide almost the same kind of information like template based approaches but using dedicated decision models. These models not only provide important attributes for each decision, but also means to relate those decisions with software architecture artifacts and other decisions. By using structured information format for each decisions and also relationship information, those models provide a better architectural overview of Software Systems. Despite the fact that model based approaches enrich the Architectural Knowledge of software systems, they are not sufficient for Enterprise Architecture. Software Architecture is only one part of Enterprise Architecture. Different Architectural decisions exist in EA that can have dependencies and relationship with artifacts and decisions from different layers of Enterprise Architecture. Our approach complement model based approaches for Software Architecture by providing more specialized attributes for EA decisions as well as more specific dependency and relationship types between EA Decisions.

5. CONCLUSION

In this paper we introduced EA Anamnesis, an approach for capturing design decisions made in Enterprise Architecture. EA Anamnesis provides a metamodel as a basis for capturing EA decisions. Based on relationship dependencies information, we also introduce a Decision dependencies representation scheme. EA Anamnesis aims to capitalize on the holistic nature of ArchiMate, by not only uncovering the business rationale behind IT decisions.

In this paper, we only provide a DSL 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 collectively come to the final architectural decision tree. How to collectively create a design decision tree, 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 manage the recording cost of our approach. Although design rationale helps architects to better understand existing designs, the main criticism is about the cost of capturing such information. As Enterprise Architecture evolves over time and increases in size and complexity the recording cost becomes higher. The return of modeling effort of our approach should be more than satisfactory in order to be applicable. To do this, effective ways of capturing design decisions during the design process should be investigated and integrated with our approach.

Acknowledgments.

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

6. REFERENCES

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

[2] J. Cummins and N. Doherty. The economics of insurance intermediaries. *Journal of Risk and Insurance*, 73(3):359–396, 2006.

[3] S. De Kinderen, K. Gaaloul, and E. Proper. Integrating value modelling into archimate. In *3rd International Conference on Exploring Service Science*, pages 54–61. IEEE, 2012.

[4] V. Haren. *Archimate 2.0 Specification*. Van Haren Publishing Series. Van Haren Publishing, 2012.

[5] J. Hoogervorst. Enterprise architecture: Enabling integration, agility and change. *International Journal of Cooperative Information Systems*, 13(03):213–233, 2004.

[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*, pages 109–120. IEEE, 2005.

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

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

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

[10] P. Louridas and P. Loucopoulos. A generic model for reflective design. *ACM Transactions on Software Engineering and Methodology*, 9(2):199–237, 2000.

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

[12] A. Ran and J. Kuusela. Design decision trees. In *Proceedings of the 8th International Workshop on Software Specification and Design*, page 172. IEEE Computer Society, 1996.

[13] 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.

[14] A. Tang, M. Babar, I. Gorton, and J. Han. A survey of architecture design rationale. *Journal of systems and software*, 79(12):1792–1804, 2006.

[15] A. Tang, Y. Jin, and J. Han. A rationale-based architecture model for design traceability and reasoning. *Journal of Systems and Software*, 80(6):918–934, 2007.

[16] J. Tyree and A. Akerman. Architecture decisions: Demystifying architecture. *Software, IEEE*, 22(2):19–27, 2005.