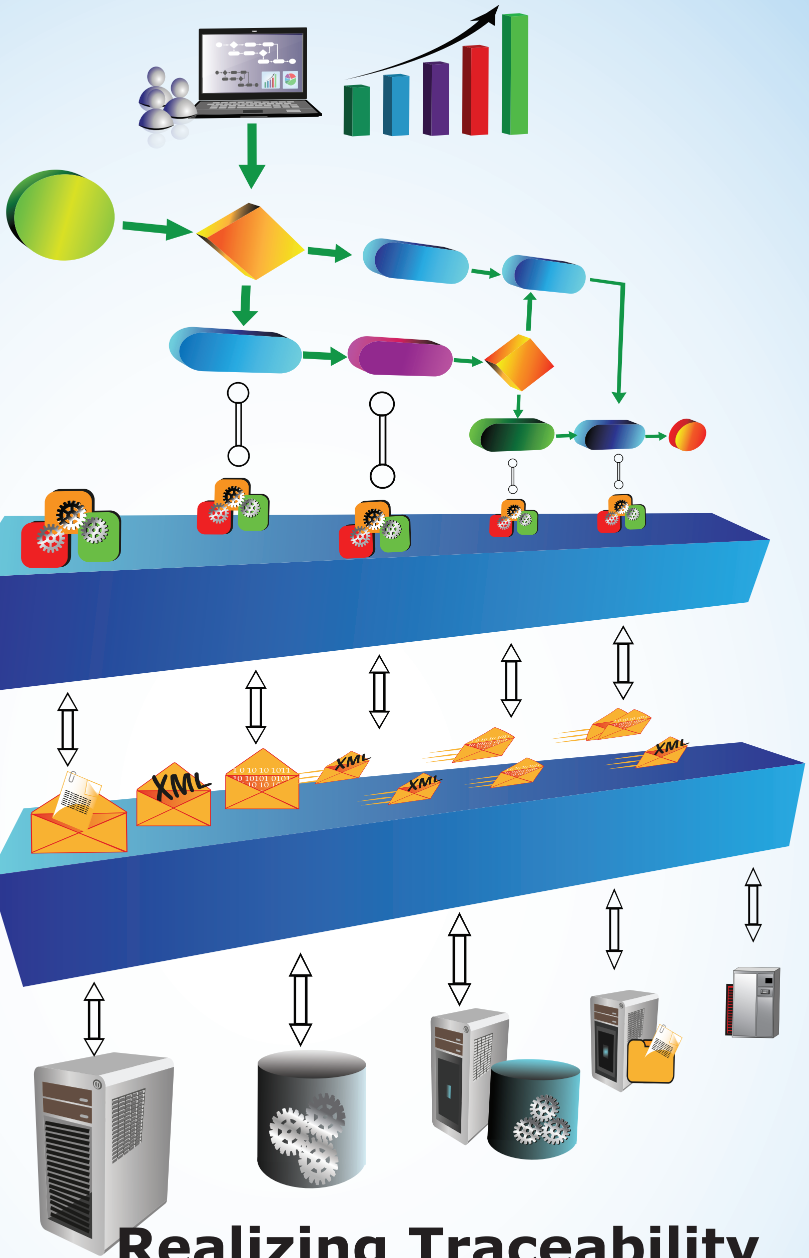


Realizing Traceability between the Enterprise Architecture and Business Value

Wilco Engelsman

REALIZING TRACEABILITY BETWEEN THE ENTERPRISE ARCHITECTURE AND BUSINESS VALUE

Wilco Engelsman

REALIZING TRACEABILITY BETWEEN THE ENTERPRISE ARCHITECTURE AND BUSINESS VALUE

DISSERTATION

to obtain
the degree of doctor at the Universiteit Twente,
on the authority of the rector magnificus,
prof. dr. ir. A. Veldkamp,
on account of the decision of the Doctorate Board
to be publicly defended
on Thursday 20 January 2022 at 16.45 hours

by

Wilco Engelsman

born on the 5th of July, 1980
in Zwolle, The Netherlands

This dissertation has been approved by:
Supervisor
prof. em. dr. R.J. Wieringa

Co-supervisor
dr. ir. M.J. van Sinderen

Cover design: shutterstock.com
Printed by: Ipskamp b.v.

ISBN: 978-90-365-5320-9
DOI: https://doi.org/10.3990/1.9789036553209
DSI: No. 22-002 ISSN 2589-7721
UNIVERSITY OF TWENTE. | DIGITAL SOCIETY INSTITUTE

Graduation Committee:

Chair / secretary:	prof. dr. J.N. Kok
Supervisor:	prof. em. dr. R.J. Wieringa
Co-supervisor:	dr. ir. M.J. van Sinderen
Committee Members:	prof. dr. M.E. Iacob
	prof. dr. G. Guizzardi
	prof. dr. O. Pastor
	prof. dr. G. Poels
	prof. dr. A. Wegmann

Realizing Traceability between the Enterprise Architecture and Business Value

Wilco Engelsman

December 26, 2021

Abstract

An enterprise architecture (EA) is a high-level representation of the enterprise. An EA is designed to realize the *business value* of an organization. First, we design and evaluate a traceability relation between ArchiMate and the business goals. We define a goal modeling language for EA, called *ARMOR*. Next, we continue our validation with practitioners and academics in terms of utility and understandability. We conclude that the concepts of ARMOR are *difficult* to understand, because the concepts are *closely* related. We propose and validate a *simplified* version of ARMOR in this thesis, with only stakeholder and goal as modeling constructs. This language and first evaluations have *influenced* the creation of the motivation layer in ArchiMate 2.0.

Second, we define a traceability relation between an e^3*value* model and an ArchiMate model. We present guidelines of how to align an ArchiMate model with an e^3*value* model. We perform a conceptual analysis of the meta-models of both languages and derive a set of *alignment hypotheses.* These are refined in an experiment with practitioners and evaluated in a case study. A key finding from this case study is that for the traceability to be useful, the *quantifications* of an e^3*value* model have to be aligned with those in an ArchiMate model.

We end this thesis with a *quantitative alignment* of an ArchiMate model with the quantifications of an e^3*value* model. We export the *economic transactions* from an e^3*value* model to an ArchiMate model as *workload requirements.* We use the ArchiMate *profiling mechanism* to store the *economic transactions* of an e^3*value* model in an ArchiMate model. The economic transactions are *propagated* throughout the ArchiMate model as workload requirements. We also annotate an ArchiMate model with *investments and expenses* and argue how these can be inserted into an e^3*value* model.

This thesis has the following *contributions.* Traceability between business goals and ArchiMate models enable us to perform *impact of change and completeness* analyses. Traceability between the e^3*value* models and ArchiMate models allows us to reason about the *economical and technological feasibility*

of a business model.

Acknowledgments

When I started my PhD-research in 2010 I never thought it would take 12 years to finish. A series of severe complications after colon surgeries ended my life as I knew it. I had to focus on getting well and had to find some sort of new stability in my life. This took the better part of a decade to achieve. After a time of careful consideration, if I would be able to finish the project I started 9 years earlier, I decided that it had to be done. I lived as a hermit for about two years, aided by covid-19 lockdowns, and steadily worked from conference deadline to conference deadline to deliver the work I needed to be able to defend my PhD-research. However, the achievement of this goal would not have been possible without the contributions of others, in some way or another.

First, I need to start with my parents, Gerrit and Wilma, for the continuous support from working myself up from the mavo to a doctorate degree. I also need to thank my sister Suzanne for providing the support I always needed.

A very special mention goes to Eelco, without his support during the time I was ill, I would not have been able to finish this project. Driving me to the hospital at a moment's notice and making sure I was all right before going home, only to return the next day. This was lifesaving. Thank you. I do not have the space to thank everyone for their support one way or another during that time, but I would like to mention (in random order) Michael, Sjoerd, Tamara, Tobias, Geert, Elmer, Rien and many others.

Roel, thank you for not only giving me the chance once, but twice to obtain a PhD. I had almost accepted that the project had come to an unfruitful end when you contacted me with the request to finish it. I also need to mention Jaap and the co-supervisor Marten. You sacrificed a lot of time when co-authoring the papers we wrote together and listening to all the half-baked ideas. Similarly, I need to thank my current employer and colleagues at Saxion for the very pleasant working environment, and in particular, Raimond, Hesther, Paul and Timber for the support and opportunity to finish my thesis. I am grateful that my previous employer BiZZdesign gave me the opportunity to start this project more than a decade ago. Although it is a decade later, I have not forgotten the colleagues of the former IS-group. It was a good place to work and the social events were always special.

To the participants from Company X from the case studies in chapter 10 and chapter 11. We did not get as far with the topic as we initially wanted, but it provided a validation for why we started this work and the case study massively improved my guidelines. Thank you.

The committee, I appreciate the time you took the read the work, the very useful feedback I received and the improvement suggestions. Thank you: Alain, Geert, Giancarlo, Maria, and Oscar. For the many others that helped me during my illness or provided any other contribution to this thesis. Thank you.

Contents

List of Figures

List of Tables

Part I

Introduction

1

Introduction

1.1 Research Motivation

Every organization exists to deliver some sort of value to its customers. This is true for both for-profit organizations and non-profit organizations. For-profit organizations sell goods or provide services to customers for money with the intention of delivering value to the customers. Selling products for money increases the value of a company. If the company does well in making a profit, the shareholders benefit. A non-profit organization creates a collective or social benefit in exchange for some sort of (monetary) contribution. These exchanges between the customers and the organization are transfers of value. A customer receives something they wish and they are willing to pay for it. Non-profit organizations therefore deliver stakeholder value instead of shareholder value. In essence these value transfers are the most important reason why an organization exists. An organization is **designed** to realize these transfers of value.

However, a focus purely on these value transfers leads to an organization that only focuses on the shareholder value of the organization and the needs of the consumer. This is not desirable. First, an organization is embedded in a society. This is where we find the external stakeholders of an organization with their **goals**, like legislators and interests groups. These impose rules and design constraints on these value exchanges. But an organization also has internal stakeholders, these all have their goals in how an organization must realize the value transactions.

The delivery of value to the customers of an organization is described in the business model [93, 49]. A **business model** abstracts away from how this value is realized internally, but only focuses on the delivery of value to the customers. In other words, a business model is only a conceptual model of how to create, deliver and capture value to the customers. It does not say anything about how this is realized [49]. An organization also needs to think about how to (internally) realize the business model. In this thesis we assume an **enterprise architecture** is used to describe the organization in more detail. Within large organizations, Enterprise Architecture (EA) is used to design an organization based on the goals it needs to support. EA is a way to design and steer an organization in such a way that it facilitates the improvement of business-IT alignment. In other words, the EA helps realizing the business value of the organization. An EA is a high level design of an organization that needs to deliver the transfers of value between economically independent actors.

The relation between business models and enterprise architecture is therefore as follows; the business model focuses on value creation to the customers

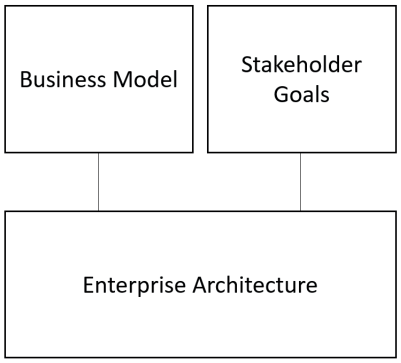


Figure 1.1: A conceptual overview of the traceability relations between the business model, stakeholder goals and the enterprise architecture in this thesis. The lines are a bidirectional traceability relation. The scope is traceability between the EA, the business model and stakeholder goals. The traceability relation between the business model and stakeholder goals is out of scope.

of an organization and provides the initial puzzle pieces. Enterprise Architecture will put these puzzle pieces together and will add design decisions how to (internally) realize the value exchanges [5]. The business model and the enterprise architecture are both based on relevant stakeholder goals.

In this thesis we will define an approach for realizing **traceability** between the **stakeholder goals**, the **business model** and the **enterprise architecture**. This chapter is structured as follows. Section 1.2 introduces the concept of business models. Section 1.3 discusses the concept of enterprise architecture and section 1.4 discusses the types of traceability we aim to achieve. We continue this chapter with a definition of the problem statement in section 1.5 and the research design in section 1.6. We conclude this chapter with the contributions of this thesis in section 1.7.

1.2 Business Models

A business model (BM) is a conceptual model of how an enterprise creates, delivers, and captures value [93, 112]. It can be described as a plan for a company to make a profit. It is not just a description of financial aspects, but a business model describes conceptual elements needed to make a profit.

For example, in the Business Model Canvas (BMC) a business model contains a description of the key partners needed to sustain a company, a description of the key activities that are needed to deliver value to the customer, a description of the products and services needed, a description of how the company can maintain the customer relations, which customers the company wants to serve and also the cost structures and revenue streams of the company [93].

Alt and Zimmerman [5] take a similar, but slightly different perspective on business models. They argue that a business model should at least contain the following elements:

- Mission
- Structure
- Processes
- Revenues
- Legal Issues
- Technology

If we look more closely at these elements, they identify goals, vision and the value proposition for the mission element of the business model. For the structure element they define actors and governance. For processes they define coordination mechanisms and for the revenues the sources of revenue and the business logic.

Legal issues have to be considered with all dimensions of the business model (i.e. they influence the general vision). These legal issues can influence the design decisions (e.g. privacy law will influence design decisions regarding the processing of personal information). Technology is both an enabler and a constraint for IT-based business models. Technological developments impact business model design, for example Spotify and Netflix would not have been possible without the current state of the internet. Therefore, according to Alt and Zimmerman [5], a business model articulates the logic, data and other evidence that supports the value proposition to a customer, combined with a structure for revenues and costs how to deliver that value.

In this thesis we take the view that the business model should represent the entire value network [126] that creates and delivers value to the customers. This includes enterprises and their stakeholders. Every business operates in a value network, so in order to be successful as an individual organization the profitability of the value network needs to be taking into account [44]. We have chosen this view because we believe that the success of a single organization is dependent on the success of other companies in the ecosystem. For example, one of the reasons Windows Mobile failed is the lack of a functioning ecosystem (e.g., app developers were unwilling to develop apps for the Windows Mobile platform or hardware developers unwilling to develop hardware based on Windows Mobile) [1] [2].

In a value network products and services are exchanged between at least two, but often more enterprises. Each enterprise focuses on its core competences and jointly they satisfy a complex customer need. Following Moore [89], we define a value network as a collection of companies that work cooperatively and competitively to satisfy customer needs. Organizations in a value network are economically independent, each must have a positive cash flow to survive. A value network can contain competitors who compete for the same customers. For example, Apple has created an extensive ecosystem for its products and services, which contains an organization responsible for taking back old Apple products for recycling. However, this company decided to cherry pick the still functioning Apple products and sell them as refurbished products, directly competing with Apple [3].

1.2.1 ***e^{3}value***

In this thesis we use *e^{3}value* to design and describe the business model of a value network [49, 44]. We have chosen *e^{3}value* because it takes an ecosystem approach and it is a conceptual modeling language. This approach consists of a lightweight graphical modeling notation and is aware of value exchanges. The notation contains elements like actor, market segment, value activity and value exchanges. *e^{3}value* allows for describing the value proposition of an organization and the activities that are required to do so. These value activities lead to a value exchange underpinned with formulas to state the amount of value exchanged.

In Figure 1.2 an educational *e^{3}value* model is presented, annotated with the name of the modeling constructs, which we discuss below. This example should be read as follows. The reader has a consumer need to read a book,

[1]https://www.zdnet.com/article/microsoft-and-windows-phone-what-went-wrong-and-where-can-they-go-from-here/

[2]https://www.zdnet.com/article/windows-10-mobile-microsoft-just-put-the-final-nail-in-the-coffin/

[3]https://appleinsider.com/articles/20/10/01/apple-sues-recycler-for-allegedly-reselling-100000-devices-it-was-hired-to-scrap

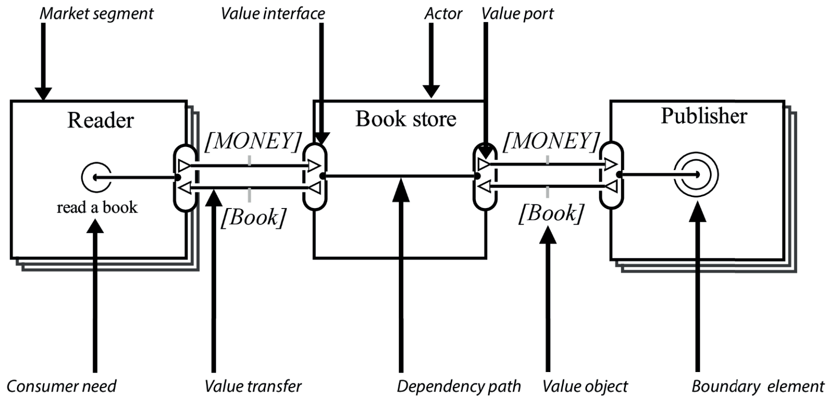


Figure 1.2: Educational e^3*value* model. This model describes the exchanges of books between a bookstore and a reader and between the book store and the publisher.

therefore the reader needs to acquire a book from the bookstore and is willing to pay money for it. In order for the bookstore to be able to sell a book, it needs to acquire the book from the publisher, also in exchange for money. The need of the customer can only be satisfied through value creation in the entire value network and is therefore dependent on the acquisition of the book by the bookstore from the publisher.

In e^3*value* an *actor* is some entity capable of performing value activities, e.g., a business, department or partner. In the example, the book store is an actor. Actors in e^3*value* are economically independent, they are responsible for creating a profit for themselves. A special case of an actor is the *market segment* (e.g., the reader or the publisher). A market segment models many actors of the same kind. In e^3*value* this means that all actors in a market segment assign economic value precisely in the same way. A *value activity* (not shown in the example) is a task performed by an actor which potentially leads to a benefit for the actor [44, 46]. In a for-profit organization a value activity can result in a positive net cash flow. In a non-profit organization a value activity should lead to a contribution to the mission of the organization, and in the case of an individual it can lead to an increase in economic utility.

A *value interface* represents what the actors offers and requests to/from its environment in terms of value objects. *Value objects* are things that are perceived by at least one actor as of economic value. A value interface consists of at least one in-going and one out-going port, through which the actor requests or offers value objects from or to its environment. The value interface models

(1) the notion of economic reciprocity and (2) bundling. *Economic reciprocity* is the idea that someone only offers something of value, or something else of higher economic value is obtained in return.

In the example, the book is exchanged for money, hence the transfers are economically reciprocal. Bundling is the case where it is only possible to offer or obtain value objects in combination (not seen in this example). Value ports between actors are connected by means of *value transfers*, which represent the willingness of actors to exchange things.

Internally in an actor there is the *dependency path*, which shows how value objects are exchanged via a value interface. For example, the sale of a book by the book store requires that this store obtains the book from a publisher. The *boundary element* of a dependency path indicates the boundary of our modeling interest. Any further transactions that take place in the real world to satisfy the consumer need are not included in our model. A *customer need* is a lack of something valuable that the actor wants to acquire. In this example, the reader wishes to read a book and acquire a book to do so.

1.2.2 The Concept of Value

Value is an important concept in this thesis and in the e^3*value* methodology. One of the goals of this thesis is to realize a traceability link between the value exchanges in e^3*value* and the internal organization realizing them. When speaking of value there are two different types of value, value-in-exchange and value-in-use [121], that are of concern to us. With e^3*value* both types are applicable. In e^3*value* goods and services can be exchanged for money (or goods). This is related to value-in-exchange, goods are exchanged on the marketplace for money or other goods. This is the classical view on value creation. A product is built and sold on the marketplace. Value is added through a production process. This is not how value is exclusively seen in e^3*value*. Value-in-use is also applicable for e^3*value*, value-in-use determines the relative value of the value-in-exchange [121]. In this case the value is the economic benefit of some good or service for an economic actor. This value depends on the increase in utility when the actor uses a good or service. Different people may assign different value to goods and services. In this case value is co-created by consumers and providers.

1.3 Enterprise Architecture

An enterprise architecture is a design or a description that makes clear the relationships between products, processes, organization, information services and technological infrastructure; it is based on a vision and on certain assumptions, principles and preferences; consists of models and underlying principles; provides frameworks and guidelines for the design and realization of products, processes, organization, information services, and technological infrastructure. It comprises a collection of simplified representations of the organization, from different viewpoints and according to the needs of stakeholders [72]). A coherent description of EA provides insight, enables communication among stakeholders and guides complicated change processes[59].

Enterprise architectures are often described with diagrams. The architects construct diagrams of products, business processes, business services, applications, infrastructure elements and the relation between these. The goal of these diagrams is to provide consistent, holistic design decisions to realize the business goals of the organization. It provides the (IT) realization projects with consistent design decisions, thus preventing that in each project different interpretations of the business goals lead to inconsistent design decisions.

These blueprints are used in the solution realization projects to serve as constraints or design scope of the solution under development [33]. By making high level design decisions early and consistent business-IT alignment can be achieved and maintained. Projects that are developed under architecture deliver more business value [106].

1.3.1 ArchiMate

In this thesis we will use ArchiMate [73] as our EA modeling language. We have chosen ArchiMate, because it is one of the most used languages [77] for EA descriptions. Also, ArchiMate has been adopted by The Open Group as a companion tool for TOGAF [114], which also will further its use in practice.

The basic idea behind ArchiMate is that it is structured around layers and aspects. At the start of this thesis ArchiMate comprised out of three layers, *the business layer, the application layer,* and *the technology layer.* The business layer models the operational organization in a technology independent manner. For example, the business services, the business processes, and the business actors. The application layer typically models the application architecture that describes the structure, behavior, and interaction of the applications of the enterprise. Finally, the technology layer models the technology architecture of the enterprise, for example the servers and network of an organization.

Each layer contains three aspects, *passive structure*, *behavior*, and *active structure*. The active structure elements are the actors in an organization, at the business layer we can identify business actors, at the application layer application components and at the technological layer nodes and devices. The aspect behavior represents the behavior performed by the business actors (e.g., business services and business processes). The aspect passive structure represents the objects on which behavior is performed (e.g., business objects, data objects). These layers and aspects are still in ArchiMate 3.1 today, it is called the ArchiMate Core framework [73].

Figure 1.3 illustrates *some* of the different aspects of an organization that can be described. We see the central notion of a *sell books* business service. We modeled the customer as an end user of the service by using the *serves* relation. The book store is responsible, through the *assignment relation*, for exposing the service to the environment. The business service is *realized* by a selling process to which a sales person is *assigned*. An application *serves* the process and the application runs on a server with Windows 10 (through the *composition relation*). Please note, that this figure is only for illustration purposes. We omitted most of the concepts and relations of ArchiMate [114].

1.4 Traceability

The main goal of this thesis is to realize traceability between the enterprise architecture and business value. In this section we will explain which kinds of traceability we aim to achieve. The first kind of traceability is traceability between a design artifact, in our case the enterprise architecture, and its requirements preceding the design. This kind of traceability is described by Ramesh [102]. This is the classical requirements engineering traceability where the problem domain (stakeholders, goals and requirements) is linked with the design realizing the requirements.

This way, there is a trace between the stakeholders, their requirements and to the design implementing the requirements, in a forward and backward manner. Forward traceability is tracing the requirement to refined requirements or the design artifact. Backward traceability is tracing from the design artifacts to its originating sources in the problem domain.

The other kind of traceability we wish to realize is found in model-driven engineering (MDE). This is where models in different phases of the design process are transformed from abstract models to more detailed models. In our case we wish to create traceability between e^3*value* models and ArchiMate [42]. In e^3*value* the focus lies on designing and evaluating the profitability

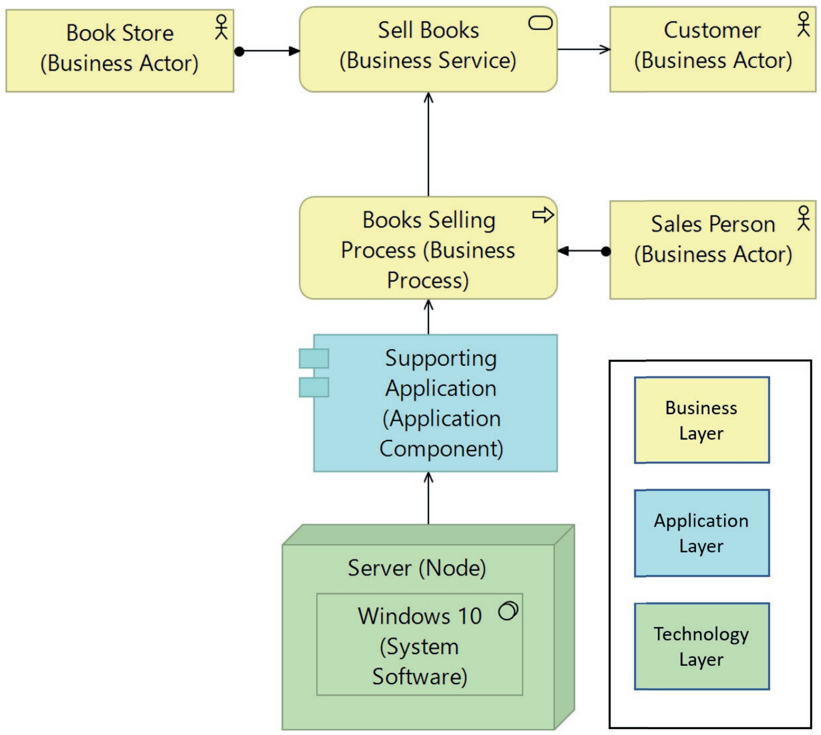

Book Store (Business Actor)
Sell Books (Business Service)
Customer (Business Actor)
Books Selling Process (Business Process)
Sales Person (Business Actor)
Supporting Application (Application Component)
Server (Node)
Windows 10 (System Software)
Business Layer
Application Layer
Technology Layer

Figure 1.3: This is a high-level educational example of an ArchiMate Model. The yellow tinted elements are business layer elements, the blue tinted elements are application layer elements and the green tinted elements are technology layer elements. Section 3.2.1 discusses ArchiMate in more detail.

of economic independent actors in a value network. ArchiMate designs the concrete organization of a single company in this value network in terms of business services, processes and IT systems.

In this thesis we aim to realize alignment between two types of models with a large difference in the level of abstraction. We therefore do not aim to realize traceability through automated formal transformations. The alignment is done using manual interventions (i.e., design decisions that need to be made).

Going from e^3*value* to ArchiMate is a vertical alignment of two different models of different levels of abstraction with requires manual interventions [82]. The main goal is alignment between e^3*value* and ArchiMate in order to reason from an e^3*value* model into an ArchiMate model and back. We do not aim to create bi-directional model transformations where both models can be transformed into each other.

1.5 Problem Statement

When we started this thesis, ArchiMate 1.0 was the standard. It did not contain the possibility to relate stakeholder goals to EA. So, there was no way to reason about the underlying goals of the enterprise architecture. This means that ArchiMate did not have the ability to show the contribution of the enterprise architecture to the business goals of the organization. This is a problem because an EA is a design artifact. An EA is therefore not a static design, it is subject to change. If we would be able to link an EA with the goals preceding its design, we could perform a change impact analysis. This would answer questions like which part of the organization needs a redesign when a business goal changes, or which stakeholders cannot be satisfied when a project or IT system fails. A second type of analysis is that of completeness analysis. Every element of the organization should be traceable to the goals preceding it.

Second, we also believe that ArchiMate should be linked to e^3*value* because this would enable us to reason about (technological) feasibility of an e^3*value* model. An e^3*value* model only focuses on the (abstract) value transfers between economically independent actors. It has no notion of how to internally realize these elements. An e^3*value* model might be feasible in terms of profitability of the actors involved, but the organization still has to deliver these transfers of value between actors. By adding traceability between ArchiMate and e^3*value* reasoning about the (technological) feasibility of the value transfers. Second, if we operationalize an actor in e^3*value* with an ArchiMate model we can also more specifically identify the investments and expenses needed to deliver the value transfers. A lack of traceability between the EA and the goals of stakeholders (problem 1 and 2) and between the BM and the EA (problem 3,4 and 5) will lead to the following problems:

1. It is impossible to perform an impact of change analysis, for example, when a business goals changes the impact on the enterprise architecture becomes impossible to assess.

2. It is hard to determine if a developed enterprise architecture is actually a complete architecture. If certain business goals or value exchanges are not satisfied by the enterprise architecture, then the architecture design is incomplete, or the business goals are not realistic.

3. It is hard to demonstrate that a developed (IT) architecture is more than just a cost center, instead of being a central component to realize the revenue of an organization.

4. It is hard to determine if a developed enterprise architecture can actually process the number of economic transactions it needs to support. In other words, without traceability, we cannot be sure whether the developed architecture will perform or whether a business model is at all technologically feasible.

5. It is hard to determine if an IT architecture is financially feasible, realizing a designed architecture requires investments and operational expenses. If this outweighs the benefits, it is not a financially feasible architecture.

Figure 1.4 elaborates the topic under investigation. We have a value network described in e^3*value*. This network contains several economic independent actors to realize some sort of customer need. Each actor has their own goals on which a business model is designed (e.g., mission and vision). To form a functional ecosystem there should be at least some sort of implicit ecosystem goals. These collective goals can conflict with the individual goals. Organizations in an ecosystem can be both competitors and allies. e^3*value* does not support goal modeling. The goals underlying an e^3*value* model are not made explicit, but they are part of e^3*value* design methodology. For pragmatic reasons we will treat these goals of an individual actor in the ecosystem as goals of the stakeholders of the EA for the remainder of this thesis.

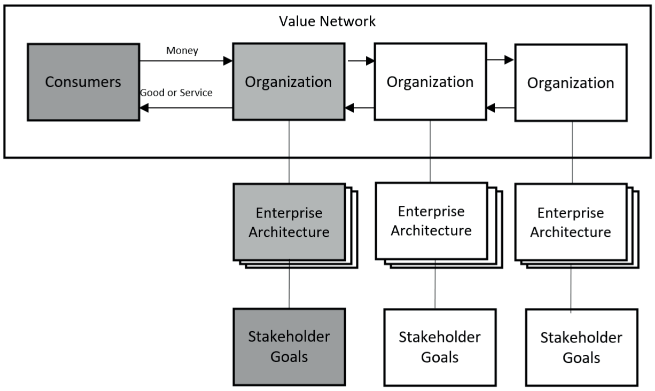


Figure 1.4: A conceptual overview of the traceability relation between the business model and the enterprise architecture realizing the business model. They grey shapes illustrate the scope of this thesis (focal company). The lines between the enterprise architecture, stakeholder goals and the value network depict a bi-directional traceability relation. The arrows are value transfers.

In this traceability relation we model the focal company, the focal company is the company for which a value network is analyzed. Each organization in this value network has its own enterprise architecture. This EA should realize the value activities of an e^3*value* model. For each e^3*value* model alternative architectures can exist, depending on the goals of the stakeholders. Therefore, our problem under investigation is as follows. We wish to investigate of how to realize traceability between a single organization of the value network, modeled in e^3*value*, the enterprise architecture, expressed in ArchiMate, and to the business goals. We have formulated the following primary design goals:

1. To design and evaluate a traceability relation between enterprise architecture and the business goals.

    - To be able to design and evaluate an enterprise architecture based on the goals of the stakeholders of the organization.
    - To be able to perform an impact of change analysis from the business goals to the enterprise architecture and back.
    - To be able to perform a completeness analysis of the enterprise architecture based on business goal realization.

   - To evaluate the traceability relation in terms of understandability of the defined concepts.

2. To design a traceability relation between the business model and the enterprise architecture.

   - To evaluate the contribution of enterprise architecture to the business model.
   - To determine if a business model is feasible in terms of financial sustainability and technological feasibility.

To do so we must realize traceability to both the business goals of an organization and the value exchanges of an organization. This traceability is not an end by itself, but a means to evaluate alternative enterprise architectures.

1.5.1 Research Questions

This research has three main research questions. The first and the third question are design questions. As discussed in the research motivation part of this thesis, an EA implements a business model and is a design artifact based on business goals. Our first research question is about realizing traceability between the business goals and EA.

- Q1: How can we extend ArchiMate with business goal modeling to realize traceability?

The second research question is a knowledge question. After constructing the goal-modeling extension we evaluated it in practice.

- Q2: How well can practitioners use this extension? Which constructs do they use correctly, which incorrectly and why?

The third research question is a design question of how to realize traceability between an e^3*value* business model and an ArchiMate EA model. This traceability is later evaluated to determine its utility. The last part of the design question is a validation question.

- Q3: How can we align an e^3*value* model with an ArchiMate model and how can we incorporate the quantifications of e^3*value* into ArchiMate?

These questions are further fleshed out in the individual chapters of this thesis. The focus of this thesis shifted during its execution. We started with extending and evaluating ArchiMate with goal modeling. This is described in the second part of this thesis. We made a design decision early that we needed elements from goal-oriented requirements engineering (GORE) and align this with ArchiMate. We believe that goal modeling for enterprise architecture is part of the same design iteration. During our validation we found out that the extension was complex and needed simplification. We end the second part with a simplification of the language.

In the third part of this thesis, we realize traceability between between e^3*value* models and ArchiMate models. We believe that business model design and enterprise architecture design are two distinct phases in organizational design, performed by different actors, which requires tooling specifically designed with these users and goals in mind. We chose to create traceability between e^3*value* and ArchiMate, because we want a separation of concerns strategy. We want to keep the value modeling in e^3*value* and the operationalization of the value model in ArchiMate. Therefore, it is not our end-goal to bring value modeling to ArchiMate, but to **align** ArchiMate with value models expressed in e^3*value*. In other words, we want to operationalize an e^3*value* model with ArchiMate by adding design decisions.

1.6 Research Design and Outline Thesis

To achieve our design goals, we follow a design science methodology [94]. This thesis follows a variant of a design science approach proposed by Wieringa [127]. Design science is the design and investigation of an artifact in context with the goal to **improve the artifact**. The artifacts we study are designed to interact with a problem context to improve something in that context, or as Van Aken [117] puts it, the scientist must develop knowledge that can help the practitioners in the field about the artifact under investigation. Design science is iterative in nature, therefore we perform repeat studies to evaluate and refine the artifacts we created.

Wieringa [127] describes that a design science project iterates over designing and investigating. Design entails to investigate a problem, design a treatment and validate this treatment. It is a cycle because it is of an iterative nature. This design cycle is part of a larger cycle, in which the results of the design cycle, a validated treatment, is transferred to the real world, used, and evaluated. The later cycle is called the engineering cycle, see figure 1.5. This chapter only describes the high-level design cycle. The individual chapters of

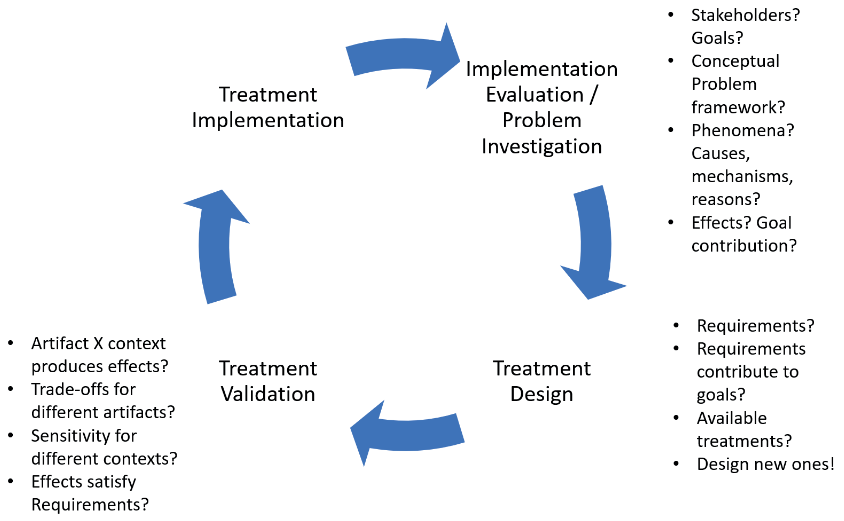

Treatment Implementation
Implementation Evaluation / Problem Investigation
Stakeholders? Goals?
Conceptual Problem framework?
Phenomena? Causes, mechanisms, reasons?
Effects? Goal contribution?
Treatment Design
Requirements?
Requirements contribute to goals?
Available treatments?
Design new ones!
Treatment Validation
Artifact X context produces effects?
Trade-offs for different artifacts?
Sensitivity for different contexts?
Effects satisfy Requirements?

Figure 1.5: An overview of the design cycle of Wieringa [127]. This thesis is structured around this cycle.

this thesis provide a more detailed description of each iteration of the design cycle. This thesis is structured around individual iterations of the engineering cycle.

The combination of the two provide a logical structure of tasks. It tells us to design a treatment we must understand the problem and justify the choice for this treatment, and it must be validated before it is implemented. We also must learn from an implementation, so we need an evaluation of the implementation.

The second part of this thesis describes the definition and primarily the evaluation of a goal-oriented requirements engineering language for enterprise architecture. We start with a definition of a language and through lab validations and transfer to practice we analyze this modeling language in terms of understandability. In each iteration we learn from the implementation. The third part of this thesis is fundamental design, but still follows the same philosophy. A treatment, a traceability relation between e^3*value* and ArchiMate, is designed and evaluated. But the evaluation is of a much lesser extent than the second part of this thesis. However, we still pursue an iterative design combined with practical cases or practitioners.

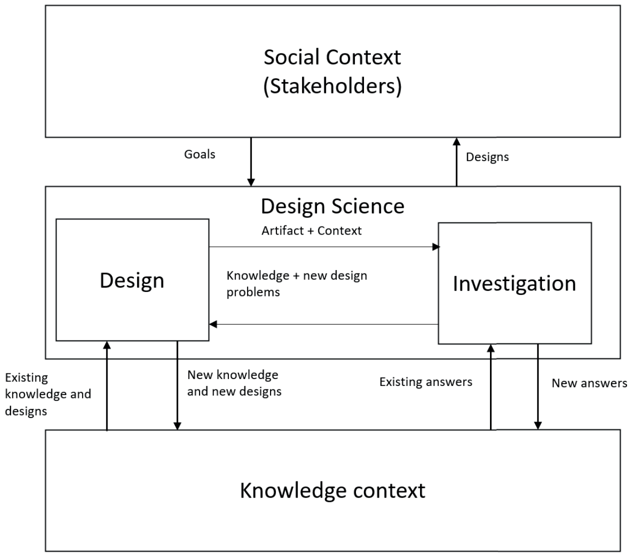


Figure 1.6: The design science framework proposed by Wieringa[127]

Figure 1.6 provides the design science framework used in this thesis. The social context of the framework contains the stakeholders of the artifact that is to be designed and evaluated. The knowledge context consists of existing theories from science, engineering, etc. The design science project uses this knowledge and may add to it by producing new designs or answering knowledge questions [127].

We will elaborate this framework with the results from our research. In general, we have two variants of this framework, applied in multiple iterations of the design cycle. For the definition and evaluation of the GORE extension to ArchiMate (called ARMOR in this thesis) the social context contains companies that use ArchiMate for their enterprise architecture design. The projected end-users of the language are enterprise architects or business analysts. Our design goals coincide with the (expected) goals of the stakeholders. Enterprise Architecture diagrams are designs, based on organizational goals and stakeholder goals. They need to show that their designs are compliant with those goals. Also, goals change over time, that would mean that the designs are possibly no longer compliant with those goal models.

Using the design cycle, we defined and evaluated an early version of ARMOR and its later derivatives. We used existing goal-oriented languages and techniques from requirements engineering and added a new language to the knowledge context [29]. During the investigation phase we investigated utility and more primarily understandability. For this investigation we used existing knowledge about GORE understandability and added to this as well [36, 37, 30].

For the third part of the thesis the social context contains companies that wish to align their ArchiMate models with e^3*value* models in order to reason about the feasibility and scalability of their business model. Direct stakeholders are enterprise architects and business modelers. The goals, we assume they have, are described in our second set of design goals. We expect that stakeholders wish to evaluate the contribution of the enterprise architecture to the business model and that they wish to determine the feasibility of a business model in terms of financial sustainability and technological feasibility. We used the design cycle to design an artifact from scratch [27, 28, 31]. We rejected existing designs [65, 64] (motivated in chapter 8) and designed guidelines for realizing traceability between e^3*value* and ArchiMate. We used existing approaches for quantitative alignment [54, 25] but had to make them concrete to work with our approach [32].

We have realized and evaluated traceability between business goals and ArchiMate EA models in four iterations of the design cycle, illustrated in table 1.1. We have realized traceability between e^3*value* business models and ArchiMate EA models in four iterations of the design cycle, illustrated in table 1.2.

Table 1.1: thesis outline and methodology for realizing and evaluating traceability between business goals and ArchiMate EA models. The references in the table refer to the articles where results of this research are published. Individual chapters are based on these articles.

Conceptual Design: we analyzed the literature on goal-oriented requirements engineering (GORE) to define an extension to ArchiMate (called ARMOR) with a **Case Study**[29].	Chapter 3
Real-World Validation and **Redesign** based on two case studies [36]. We tested our approach at a governmental institute for pension payments and at a large drinking water provider in the Netherlands.	Chapter 4
Real-World Validation where we investigated the understandability of ARMOR by enterprise architects. We tested the understandability of ARMOR during official ArchiMate 2.0 certified courses [30].	Chapter 5
A **Literature Review** of experiments about understandability issues of conceptual modeling languages. The results are used to refine our last experiment.	Chapter 6
Real-World Validation where we investigated the understandability of ARMOR by academics during a workshop at the REFSQ conference and we provided a final **Redesign** of ARMOR [37]. This chapter is an iteration of the design cycle where we perform a *sensitivity analysis*. We changed the level of education of the users of the language.	Chapter 7

Table 1.2: thesis outline and methodology for realizing traceability between e^3*value* and ArchiMate. The references in the table refer to the articles where results of this research are published. Individual chapters are directly based on these articles.

Conceptual Design: We analyzed the meta-models of e^3*value* and ArchiMate to define an initial version of the guidelines (version 1). We tested it on a small real-world example: an EA for the Cirque du Soleil [27]. This is a first iteration of the design cycle.	Chapter 8
Lab Validation and Redesign: We tested the guidelines in an experiment where we compared the EAs designed by practitioners from a business model in a laboratory assignment, with the EA that results from our application of the guidelines [28]. Although the assignment took place in the lab, the cases for which the practitioners designed an e^3*value* model and an EA were from the real world: the companies where they were employed. The analysis of the models from this experiment led to a redesign of the guidelines (version 2).	Chapter 9
Real-World Validation and Redesign: We applied the guidelines to a real-world case to redesign the business layer of the enterprise architecture of an enterprise. This experience led to a further improvement of the guidelines (version 3) [31].	Chapter 10
Conceptual Design and an **Initial Field Validation** of the quantification of ArchiMate. After an initial validation in practice of our guidelines we had to perform an additional conceptual analysis to import the quantifications of e^3*value* into ArchiMate and to extend ArchiMate with investments and expenses. We propagate the economic transactions from e^3*value* over an ArchiMate model as workload requirements. We then collect these investments and expenses for the architecture and insert them into an e^3*value* business model. We validated this idea with an initial field validation [32].	Chapter 11

This thesis is a bundle of published articles. We added an introduction and conclusion based on these articles and removed the related work from each individual paper and placed it in a separate chapter. This thesis has the following outline. Chapter 2 introduces the theoretical foundations, based on the related work from the original articles. We investigate related work in realizing traceability between business goals and enterprise architecture, understandability of GORE, and between business models and enterprise architecture. Chapter 3 introduces the goal modeling extension for ArchiMate and provides an initial evaluation through a case study and answers Q1. Chapters 4, 5, and 7 discuss three validations of this extension in practice and answer Q2. Chapter 4 focuses on utility and understandability of ARMOR, chapters 5 and 7 focus mainly on understandability of ARMOR. Using our results from these validations, we also provide a redesign of the goal modeling aspects in chapter 7. Chapter 6 discusses a literature review regarding understandability of graphical modeling notions of GORE and UML. We identify theories and explanations of why these languages are so difficult to understand and perform our last experiment based on the conclusions of this review.

Chapter 8 provides us with an initial design of the traceability between an e^3*value* model and an ArchiMate model. We extend this work in chapter 9 by analyzing different e^3*value* models and we introduce guidelines that would help practitioners in creating their own model alignments. In chapter 10 we evaluate and improve these guidelines. Based on this validation we introduce a quantification of an ArchiMate model in chapter 11 with economic transactions, investments and expenses. Chapters 8, 9, 10 and 11 answer Q3. We conclude this thesis in chapter 12 with an evaluation of our design goals and we provide a research outlook for future research based on the results of this thesis.

1.7 Contribution

The main contribution of this thesis is that it provides multiple ways to evaluate alternative enterprise architectures. This is realized by defining and evaluating a goal modeling extension to ArchiMate. This extension realizes traceability between the stakeholders, business goals and the enterprise architecture. The third part of the thesis realizes traceability between e^3*value* models and ArchiMate models, in order to evaluate the technological and financial feasibility of a business model.

The first problem this thesis solves is the traceability between the business goals and the enterprise architecture. It extends and evaluates ArchiMate with goal modeling to realize traceability between business goals and enterprise architecture. This allows us to perform impact of change and completeness analyses of the enterprise architecture. Evaluation of alternatives is made more concrete as well. This extension is then evaluated multiple times in practice. During these case studies we interviewed enterprise architects or evaluated their constructed models to determine the utility and understandability.

Finally, we introduce a redesigned and extremely simplified goal-modeling extension for ArchiMate.

The evaluation of ARMOR also resulted in an evaluation of how understandable GORE constructs are. Our language is based on similar languages found in the literature. We identified which GORE constructs are understood best and which poorly.

The third part of this thesis realizes traceability between e^3*value* models and ArchiMate models. We realize traceability between e^3*value* and ArchiMate and use this traceability to operationalize the business model with ArchiMate diagrams. We use this traceability to evaluate if an enterprise architecture can support the economic transactions of an EA and we enrich ArchiMate models with financial quantifications and insert these into an e^3*value* model to evaluate the financial and technological feasibility of a business model. We define workload requirements based on the economic transactions in e^3*value* and propagate these over the ArchiMate model. We extend ArchiMate with attributes for fixed expenses, variable expenses and investments and insert these into an e^3*value* model.

2

Related Work

2.1 Introduction

The topics of our research are goal-oriented requirements engineering (GORE), understandability of GORE, Business Models (BM) and the relation to Enterprise Architecture (EA). In order to position and scope our work we investigated the literature for related work. Section 2.2 discusses related work for goal-modeling, section 2.3 discusses related work regarding understandability of goal modeling and finally section 2.4 introduces and discusses related work for business models related to EA and ArchiMate.

2.2 Related Work for Goal Modeling

Requirements management plays a central role in TOGAF's ADM [115]. TOGAF provides a limited set of guidelines for the elicitation, documentation and management of requirements, primarily by referring to external sources. TOGAF's content meta-model, part of the content framework, defines a number of concepts related to requirements and business motivation; however, this part has been worked out in little detail compared to other parts of the content meta-model, and the relation with other domains is weak. Also, the content framework does not propose a notation for the concepts.

The Integrated Architecture Framework (IAF) is Cap Gemini's architectural framework [120]. Like TOGAF, this framework also recognizes the importance of requirements for EAs. IAF recognizes requirements at both the contextual and conceptual level. At the contextual level they identify business requirements that answer the why question and at the conceptual level they provide more detailed requirements. But IAF lacks a detailed description of how to represent either business requirements or the more detailed requirements. It mainly lacks concept definition and a requirements language to represent the requirements.

Clements & Bass extend software architecture modeling with Goal Oriented Requirements Engineering (GORE), but remove all notational conventions of GORE techniques and return to a classic bulleted list of possible goals and stakeholders [21]. This makes goal-oriented modeling usable for requirements and architecture engineering workshops with practitioners, but does not help to support the kinds of analysis that we mentioned earlier in the introduction part of this thesis.

Stirna et al. describe an approach to enterprise modeling that includes linking goals to enterprise models [109]. However they do not describe concrete modeling notations that are needed to extend existing EA modeling techniques.

Jureta and Faulkner [60] sketch a goal-oriented language that links goals and a number of other intentional structures to actors, but not to EA models. Horkhoff and Yu present a method to evaluate the achievement of goals by enterprise models, all represented in i* [52].

i* has also been used as a problem investigation technique for architecture design and business modeling [128]. This way the motivation for architectural elements is linked to their implementation. Yu et al. [128] illustrate the potential benefit of using BMM and i* in combination to support intentional modeling and analysis of EAs. This work does not consider the integration or alignment of these languages with existing enterprise modeling languages. Gordijn et al. [48] extends intentional modeling with value modeling, by combining the i* framework and the e^3*value* methodology.

Braun and Winter [12] discuss relevant meta-models for EAs. They introduce three layers for EA modeling, the strategy layer, the organizational layer and the application layer. This work is mostly relevant for ArchiMate, but they do introduce relevant concepts. These concepts are found in the strategy layer. The strategy layer models the business units, services, goals and their performance indicators. However, it maps mostly to the ArchiMate business layer whereas they introduce services and business units. Their notion of a goal is also limited to only the strategy layer, whereas we identify goals for the business layer, application layer and technology layer. We also introduce ways how to relate goals to the behavioral elements in the EA, in such a way that already exists in existing RE languages like KAOS [23]. We also introduce modeling concepts, which Braun and Winter [12] lacks.

Kurpjuweit and Winter [67] and Lagerstrom et al. [70] propose a way of working similar to the modeling concepts introduced in this thesis. They take both a stakeholder and goal-oriented approach to derive architectural models based on both stakeholder needs and causal effect relations, which is also similar to an approach we suggested here [34]. Our work supplements this work with a modeling language that can be used to capture these results and explicitly show how the derived models implement the stakeholder needs and goals. Our work has a better grounding in GORE theory, thus supporting more concrete concepts and relations, like a distinction between goals and requirements, and a distinction between means-end, decomposition, conflict and contribution.

Concerning tool support for EA, many tools claim to support requirements modeling (e.g. System Architect and Powerdesigner). However, this support is often limited to the documentation of requirements as structured lists, or the modeling of use cases. Furthermore, they do not offer graphical modeling techniques, nor the integration with other modeling domains.

Design and Engineering Methodology for Organizations (DEMO) [26] is a methodology for the design, engineering, and implementation of organizations and networks of organizations Originally meant as an RE approach for information systems, however the authors quickly found out that it was also applicable for business process engineering and workflow management. DEMO is used to not only specify organizations at a behavior level, but also at the component level, trying to bridge the gap between behavior specification and design specification DEMO focuses on specifying the essential models of an organization The approach is well validated and growing in popularity.

Our work takes elements from GORE and applies it to the field of enterprise architecture. We introduce techniques to record the business goals of an organization and create traceability between the stakeholders, goals and the enterprise architecture. This traceability enables us to perform an *impact of change* analysis in both a top-down and bottom-up manner. We can also perform a *completeness analysis* of the enterprise architecture, in which gaps between the goals of the organization and the enterprise architecture can be used to evaluate the completeness of the design. Finally, using GORE analysis techniques we can systematically bridge the gap between the problem domain and the solution domain and aid the enterprise architecture design process.

2.3 Related Work For Understandability GORE

Goal-oriented requirements engineering (GORE) modeling languages have been around for almost thirty years [23, 130]. However, transfer to practice so far has been very limited [79]. The leading notations are KAOS and i*. Previous research showed that these GORE languages are very rich in notational concepts and therefore possibly difficult to understand.

Moody et al.[88] identified improvements for i* and validated the constructs of i* in practice, based on Moody's theory of notions [87]. Moody et al. [85, 86] identified many opportunities for clarification and simplification of the i* notation. This contrasts with our work, since we will not propose an update of the visuals.

Caire et al. [19] also investigated the understandability of i*. They focused on the ease of understanding of a concept by asking subjects to infer its definition by its visual representation. They had novices design a new icon set for i* and validated these icons in a new case study. This contrasts with our work because they focus on notations and we focus on concepts.

Carvallo & Franch [20] provided an experience report about the use of i* in architecting hybrid systems. They concluded that i* could be used for this purpose for stakeholders and modelers, provided that i* was simplified. Our work extends on these findings. We also found out that related concepts are hard to distinguish (i.e the distinction between driver,assessment,goal, the distinction between requirement and goal and the distinction between decomposition and influence).

Matulevičius & Heymans [79] compared i* and KAOS to determine which language was more understandable. The relevant conclusions for this work were that the GORE languages had ill defined constructs and were there hard to use, GORE languages also lacked methodological guidelines to assist users in using the languages. i* and KAOS contain constructs not used in practice and contain different constructs representing the same thing. After an ontological analysis they concluded that the i* goal and soft goal are essentially the same concept, just as the means-end relation and the contribution relation [79]. Carvallo et al [20] recommended that practitioners should not and need not learn the entire syntax of i*. These conclusions were also found in our work.

The literature only provides a few explanations why GORE languages are so hard to understand.

The first explanation is that of experience: The more experience a user has, the higher the ability to understand the language. This was found by Hadar et al, [50], Soh et al [107], and Cruz et al [22]. This explanation goes both ways: Someone who does not understand a notation will avoid using the notation, and if he or she uses it, will stop using it very soon. Conversely, repeated use of a notation that one understand reasonably well, will produce increased understanding.

The second explanation authors give is is that there are design flaws with the languages, e.g. are too many concepts, semantic definitions are unclear, or the visualization of the constructs is ill designed. This was found by Matulevicus and Heymans [79], Purchase et al [100] and Siau and Loo [105]. This was also the explanation we used in [36] and [30].

The third explanation given in the literature provides is is that supporting materials are badly designed. Several authors mention that languages are plagued with bad tooling and bad training material [20], [79] and [105].

Our work builds and adds on this. We investigate and evaluate an active understanding of GORE concepts by practitioners and academics. We evaluate understandability of well known concepts found in GORE literature. We provide (hypothetical) explanations regarding the underlying understandability issues. Our explanations are in line with an ontological analysis of the motivation layer of ArchiMate [8].

Since our language is based on existing GORE languages our results should apply to those languages as well. During the evaluation process initial results from were used to simplify the motivation part of ArchiMate in some part. The difference between soft goal and hard goal was removed and the conflict relation between goals was replaced by a double negative influence.

Another contrast is that most of the empirical studies of the usability of GORE languages have been done with students, while we do our empirical studies mostly with practitioners and academics.

2.4 Related Work For Business Models

There is some existing work done trying to link business models, goal modeling and EA. The topic of this thesis is in essence realizing traceability from the EA to business context. In previous work we were involved in extending ArchiMate with goal-oriented concepts [101, 29] to enable goal modeling and reasoning about the contribution of the EA to the business goals of the organization. Related to this is the work of Iacob et al. [56] where they propose a method for IT portfolio evaluation using ArchiMate and the motivation extension. Aldea et al. also propose a way to link EA to the business strategy of the organization [4]. The difference with our work is that this more linked to organizational goals than business models. But it is related in such a way that it realizes traceability to be able to perform different kinds of analysis.

De Kinderen, Gaaloul and Proper [65] propose to link ArchiMate to e^3*value* using DEMO [26]. They do not propose a direct mapping between ArchiMate and e^3*value*. They wish to introduce transactionality in ArchiMate by using the DEMO language as an intermediary language [26]. They do not use the realized traceability for quantitative alignment. We use the economic transactions from e^3*value* as workload requirements for IT systems and quantify the ArchiMate model with investments and expenses to identify the investments and expenses and insert them into the e^3*value* model for Net Present Value (NPV) calculations. We also do not agree with the mappings they made between e^3*value* and ArchiMate. We will elaborate on this in chapter 8.

Janssen and Gordijn directly map ArchiMate to e^3*value* but they use a different approach, which according to us, is incorrect. They link value activities to business processes instead of business services. We believe the business service is the correct mapping to a value activity. We will elaborate this in chapter 8. Also, they determine much less instances of possible mappings. But they do agree that the business layer of ArchiMate should be mapped to e^3*value* directly [17].

Gordijn et al. [48] propose a method to combine i* with e^3*value* with no focus realizing on traceability. Andersson et al. [6] describe the alignment of business models and goals. They have developed templates that align goal statements with value propositions.

Meertens et al. [81] propose similar work, but instead of using e^3*value* they provide a mapping from the Business Model Canvas to ArchiMate. Pessoa et al. [95] developed a method for requirements elicitation for business models using an early version of the motivation extension of ArchiMate.

Gordijn et al. [47] propose a method for requirements engineering for e-services. Aldea et al. [3] propose adaptations of ArchiMate to incorporate value modeling, but does not try to create traceability between different languages and the concepts introduced are less detailed than those using e^3*value*. In general, the major difference of our work with related work is our focus on traceability through models.

Gordijn et al [47] propose a method for requirements engineering for e-services. In this work they take the requirements engineering perspective to design an e-service. They identify different viewpoints and design a service using WSDL and BPEL. This work is of a completely different scope than ours. We stay at a higher level of abstraction to answer different questions. We wish to link the BM and the EA. De Kinderen, Gaaloul and Proper propose to link ArchiMate to e^3*value* using an intermediary language (DEMO [26]). They do not propose a direct mapping [65].

Petrikina et al. [96] describe a preliminary investigation about linking business models with EA at the meta-model level. The authors propose to link the business model to the products and services and create a new meta-model. However, this work is preliminary and they have not identified any alignment guidelines.

The most recent relevant work is that by The Open Group [123]. They incorporate additional new concepts in ArchiMate, based on the business model canvas (BMC). Meertens et al. propose similar work, but instead of using e^3*value* they provide a mapping from the Business Model Canvas (BMC) to ArchiMate [81, 55].

Also relevant is the work of Fritscher and Pigneur [41, 40]. They link EA with business models with the BMC as well, but on a very coarse grained level. They do not realize actual traceability to different concepts of different languages nor do they provide guidelines or building blocks. Aldea et al. propose adaptations of ArchiMate to incorporate value modeling [3], but do not try to create traceability between different languages. They also used different concepts to expose the value to the environment (i.e business processes instead of services). We believe that adding more concepts to ArchiMate is not the solution. Adding traceability between the different models (and users) would allow for the same reasoning, without making the language cognitive harder to understand. Also, we believe that business models that do not take the entire value network in to account are of limited use in the future [126, 80].

Iacob et al. [55, 81] propose a mapping from the Business Model Canvas (BMC) [93] to ArchiMate. Since the BMC is oriented towards the *single* enterprise, this work misses the networked ecosystem point of view that is crucial to most ecosystems. We claim that exploration of the ecosystem, e.g. all participating actors and the ICT systems, need to be included in business model analysis, rather than just a single enterprise and its direct customers and supplier. Moreover, the BMC does not have the capability to quantify the business model and simulate market scenarios, as e^3*value* has, nor does the BMC have the capability to quantify ArchiMate and bring this quantification to a business model expressed in e^3*value*.

Recently The Open Group also proposed to incorporate business modeling concepts in ArchiMate [123]. The result is a version of ArchiMate that has even more symbols and concepts than it has now. And it does not solve the problem of traceability between business models and enterprise architectures, because guidelines are absent.

An application of graph-based semantic techniques to specify, integrate and analyze multiple, heterogeneous enterprise models is explored by Caetano et al. [18]. They use e^3*value*, ArchiMate and the BMC. The difference with our work is that we focus mainly on guidelines for practitioners.

Derzi et al. [25] realize traceability between UML deployment diagrams and e^3*value*. They annotate UML diagrams with investments and expenses and create traceability between UML and e^3*value* to be able analyze the profitability of an organization with the proposed IT. Deployment diagrams are used because they indicate ownership of ICT components, and ownership comes with an investment and operational expenses. These financials are important for the e^3*value* business model. Our work shares some similarities, we take the basic idea, but extend on this. We realize bi-directional traceability. We import economic transactions into ArchiMate for scalability reasoning in conjunction with aggregating investments and expenses from ArchiMate into e^3*value*. Our solution also has more semantics, which can be used to create tool support. el.

Iacob and Jonkers introduce a generic quantification approach for ArchiMate [54]. They describe a generic approach of how to perform performance analysis using workload and response times on an ArchiMate model. Our work is based on the same principles, we derive our performance requirements from e^3*value* and we quantify ArchiMate with investments and expenses. Obviously, the work of Iacob and Jonkers is restricted to ArchiMate only and therefore does not include a networked business model point of view.

Miguens [83] proposes to introduce an additional viewpoint for ArchiMate where investment information can be assigned and calculated. We do not want to perform actual investment calculations in ArchiMate beyond aggregating the information. We do all the calculations in e^3*value* because they are part of business model analysis. Miguens also does not take the business ecosystem perspective as we do, nor do they have a way to identify performance requirements based on economic transactions.

Summarizing, the major difference of our work with related work is our focus on the value network, traceability, extended with alignment guidelines, and building blocks to construct EA models based on the BM.

Our approach differs from others because we use a networked approach to business models, allow quantification of business models, and define and test traceability guidelines to align business models with an enterprise architecture.

We are also able to export the quantifications of e^3*value* to ArchiMate. The economic transactions in e^3*value* are brought to ArchiMate models and we enrich ArchiMate models with investments and expenses and can insert these back into an e^3*value* model. Our quantitative alignment between e^3*value* and ArchiMate allows us to reason about if IT can support the economic transactions found in the business model, thus providing a form of scenario analysis where the feasibility of a proposed business model is tested in terms of technological support. Quantification of ArchiMate with investments and expenses allows us to identify investments and costs in IT and insert them into an e^3*value* model and evaluate the financial feasibility of a business model.

Part II

Goal Modeling

3

Extending ArchiMate: ARMOR[1]

[1]This chapter is based on an the article in the Journal of Enterprise Information Systems [29]

3.1 Introduction

Requirements modeling is an important activity in the process of designing and managing enterprise architectures (EAs). Brooks (1986) mentions [15], ‘No other part of the work so cripples the resulting system if done wrong’. This quote refers to the design of software architectures, but applies as well and maybe even more so to the elicitation and analysis of the requirements that should be addressed by enterprise architecture design.

Nonetheless, most EA modeling techniques focus on what the enterprise should do by representing ‘as-is’ and ‘to-be’ architectures in terms of informational, behavioral and structural model elements at different architectural layers, e.g., a business, application and technology layer. When we started this research, little or no attention was paid to represent (explicitly) the motivations or rationale, i.e. the why, behind the architectures in terms of goals and requirements.

In contrast to EA modeling techniques, methods for EA, such as The Open Group Architecture Framework [115], acknowledge that goals and requirements are central drivers for the architecture development process. In TOGAF’s architecture development method (ADM), requirements management is a central process that applies to all phases of the ADM cycle. The ability to deal with changing requirements is crucial to the ADM, since architecture by its very nature deals with uncertainty and change, bridging the divide between the aspirations of the stakeholders and what can be delivered as a practical solution.

Requirements modeling helps to understand, structure and analyze the way business requirements are related to information technology (IT) requirements, and vice versa, thereby facilitating business–IT alignment. For example, the concept of ‘goal’ in goal-oriented requirements modeling is used to define some intended effect that is desired by some stakeholder, i.e. what should be achieved. This goal may be related to more abstract (business) goals that define why the goal is needed, and may also be related to more concrete (IT) goals that defines how the goal can be realized.

The explicit definition of these relations facilitates traceability among the motivations and concerns of stakeholders, their goals and the (design) artifacts that ultimately realize the goals. Goals have to be refined into requirements before their realization can be assigned to some artifact, such as a business service, business process, application service or application component. When talking about requirements modeling, we mean the modeling of goals and requirements. For now, goals can be considered as abstract requirements that need to be made more concrete before they can be realized by elements of the EA. The explicit modeling of the motivation underlying EAs using goals enables new types of analysis from the requirements engineering (RE) domain. For example, one can analyze to what extent the EA meets the stakeholder's goals, whether these goals may conflict, the impact of revised goals on the enterprise, and vice versa.

Furthermore, alternative architectures may be assessed based on their ability to meet stakeholder goals. In this chapter, we assume that ArchiMate 1.0 [73]) is used for EA modeling.

The purpose of this chapter is to define a language, called ARMOR, for modeling the motivation of EAs in terms of goals and requirements and to provide an initial validation of the language based on a case study. This language is aligned with the ArchiMate language. Furthermore, we illustrate the use of ARMOR for analyzing EAs, while focusing on business–IT alignment issues.

The remainder of this chapter is structured as follows. Section 3.2 describes the ArchiMate modeling framework and its extension towards motivation modeling. Section introduces the language requirements 3.3. Section 3.4 presents ARMOR in terms of its concepts and their notations. Section 3.5 illustrates ARMOR by means of a real-life example. Section 3.6 provides an overview of the available analysis techniques and tool support for ARMOR. Section 3.7 describes preliminary work into the integration of principles into ARMOR and finally section 3.8 presents our conclusions.

3.2 Enterprise Architecture

EA is a design or a description that makes clear the relationships between products, processes, organization, information services and technological infrastructure; it is based on a vision and on certain assumptions, principles and preferences; consists of models and underlying principles; provides frameworks and guidelines for the design and realization of products, processes, organization, information services, and technological infrastructure.

It comprises a collection of simplified representations of the organization, from different viewpoints and according to the needs of different stakeholders [72]). A coherent description of EA provides insight, enables communication among stakeholders and guides complicated change processes[59]. ArchiMate is an open standard of The Open Group, provides a language to create such descriptions in a precise and formal way. ArchiMate defines concepts for describing architectures at the business, application, and technology layers, as well as the relationships between these layers. Thus, it addresses the ubiqui-

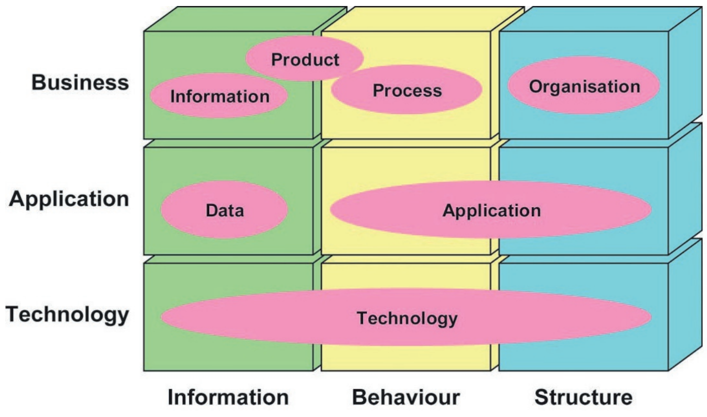


Figure 3.1: The original ArchiMate framework.

tous problem of business–IT alignment. ArchiMate originally results from a public/ private research project, a cooperation of companies, universities and research institutes.

3.2.1 ArchiMate framework

Figure 3.1 depicts the modeling framework that underlies the ArchiMate language [72]. This framework decomposes an enterprise along two dimensions: layers, which represent successive abstraction levels at which an enterprise is modeled, and aspects, which represent different concerns of the enterprise that need to be modeled. The layer dimension distinguishes three main layers:

- business layer, which offers products and services to external customers that are realized in the organization by business processes;
- application layer, which supports the business layer with application services that are realized by (software) application components;
- technology layer, which offers infrastructural services (e.g. processing, storage and communication services) that are needed to run applications, and are realized by computer and communication devices and system software.

The aspect dimension distinguishes the following modeling aspects:

- structure aspect, which represents the actors (systems, components, people, departments, etc.) involved and how they are related;
- behavior aspect, which represents the behavior (e.g. processes and services) that is performed by the actors, and the way the actors interact;
- information aspect, which represents the problem domain knowledge that is used by and communicated between the actors through their behaviors.

The structuring into dimensions allows one to model an enterprise from different viewpoints, where a viewpoint [57] is characterized by one's position along each dimension. A viewpoint represents a certain perspective on the enterprise that is of interest to one or more stakeholders. A stakeholder typically focuses on a (small) range along each of the dimensions. The intersection of these ranges spans a viewpoint. For example, each cube in figure 3.1 represents the intersection of a single layer and single aspect. A viewpoint may span multiple or only part of a layer or aspect. Furthermore, depending on the choice of viewpoints, they may (and often will) overlap. Each viewpoint comprises a number of concepts that are used to model an EA covering the levels of abstraction and aspects represented by that viewpoint. Accordingly, overlapping viewpoints may comprise overlapping concepts. In order to define, maintain and apply concepts for EA modeling in a structured and consistent way, these concepts are organized in orthogonal, i.e. non-overlapping 'viewpoints', called domains. Each domain represents a set of concepts that is used to model systems from a particular viewpoint. The ellipses in figure 3.1 represent common modeling domains that have been defined for ArchiMate.

3.2.2 Extended Framework

To support the modeling of intentional properties an extension of the ArchiMate framework is proposed, as depicted in figure 3.2. This extension consists of the motivation aspect, which resembles the motivation (or why) column of the Zachman framework [131].

The value layer represents the value of the services and products that are offered to customers. For example, existing work on value modeling, such as value modeling with e^3*value*, can be positioned in this layer. The 'Value' concept of ArchiMate fits in this layer, and could be extended to model specific types of value, such as cost, and networks of value exchanges.

ArchiMate does not provide any concepts for modeling the motivation aspect. The remainder of this section introduces three modeling domains within the motivation aspect: the stakeholder domain, the principles domain and the requirements domain.

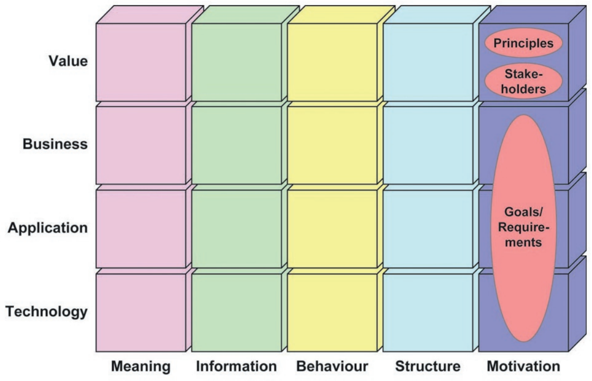


Figure 3.2: The ArchiMate framework extended with a motivation column and a value layer. This framework was proposed in the original articles [101, 29] and became irrelevant. It has been replaced in many iterations during the development of ArchiMate. The motivation column can still be found in the latest version of ArchiMate. The new strategy layer replaced the value layer [114].

Stakeholder domain

This domain models the stakeholders of the enterprise, including their concerns and the assessment of these concerns. A concern is interpreted as some area of attention or interest. For example, a CEO may be concerned with executing the mission of the enterprise, a CIO with the clarity of the EA and its ability to adapt to change, and a system's manager with the capacity and reliability of the computing and networking platforms used within the enterprise. These concerns may be assessed using a strengths, weaknesses, opportunities and threats (SWOT) analysis. For example, this analysis may reveal that the enterprise's architecture lacks traceability, which makes it difficult to change.

In addition, the users or customers of the enterprise may be considered as stakeholders. Customers may be concerned with e.g. the diversity of the products and services that are offered or the privacy of their information. Also these concerns may be assessed (not necessarily in terms of SWOT) to reveal customer needs.

Requirements Domain

This domain models the goals, requirements and expectations that constrain the design of the EA. These goals, requirements and expectations typically originate from the assessment of concerns in the stakeholders domain. This assessment may reveal strengths, weaknesses, opportunities or threats that need to be addressed by changing existing goals or setting new ones.

Principles domain

This domain models the architectural principles used in architecture design. Principles are normative restrictions on the EA. The principles use business drivers of the organization, found in the stakeholder domain and requirements domain. The current version of the domain merely identifies the need for such a domain. Section 3.7 will provide more information about the possible integration of principles into ARMOR.

3.3 Requirements Modeling

Besides its alignment to ArchiMate, we want the ARMOR language to align with concepts and ideas from existing languages for requirements modeling, wherever possible. Our intention is not to introduce a new language per se, but one that meets our modeling requirements. These requirements are described first, followed by an overview of the following techniques for goal modeling: the Business Motivation Model [16], the i* framework [129], and the KAOS notation from Dardenne [23].

3.3.1 Language Requirements

1. Re-use of concepts and ideas from existing languages and methods for goal modeling.
2. Alignment with ArchiMate.
3. Traceability. Adaptation to change is an important requirement for EAs. In order to support impact of change analysis, abstract goals should be traceable to the more concrete goals and design artifacts such as services and processes that implement these abstract goals; and vice versa.

4. Facilitate documentation, communication and reasoning about requirements.

5. KISS (keep it small and simple). ARMOR should be based on a small set of generic concepts that allows one to model the motivation aspect of EAs in an intuitive way. This should also help in obtaining a language that is easy to learn, understand and apply.

6. Extensible. It should be possible to extend ARMOR with specialized concepts and associated analysis techniques. This would allow users to choose between basic and advanced versions of ARMOR.

Requirements 4–6 are considered as 'soft' requirements. In practice, this means that they are used as guidelines for making decisions rather than hard criteria with predefined norms that can be validated afterwards. Our goal is to capture the business context leading to EA, in other words to capture the high-level goals that lead to the architectural designs. GORE facilitates capturing the business context [118] and provides a structured way to refine these goals into requirements that can be assigned to elements from the architecture. GORE itself also provides a number of advantages [118], like improved traceability, evaluation of alternatives and reasoning about conflicts. We opted for a goal-oriented RE language, based upon these goals and possible advantages. Therefore, the subsequent sections will only focus on goal oriented RE languages.

3.3.2 Business Motivation Model

BMM provides a structure of concepts for developing, communicating and managing business plans. The concepts can be used to model (i) the factors that motivate a business plan, (ii) the elements that constitute the business plan and (iii) the relationships between these factors and elements [16]. The central notion of the BMM is motivation. An enterprise should not only define in its business plan what approach it follows for its business activities, but also why it follows this approach and what results it wants to achieve. Figure 3.3 depicts an overview of the BMM. The following three major parts are distinguished:

1. Ends, which describe the aspirations of the enterprise, i.e. what the enterprise wants to accomplish;

2. Means, which describe the action plans of the enterprise to achieve the ends, and the capabilities that can be exploited for this purpose;

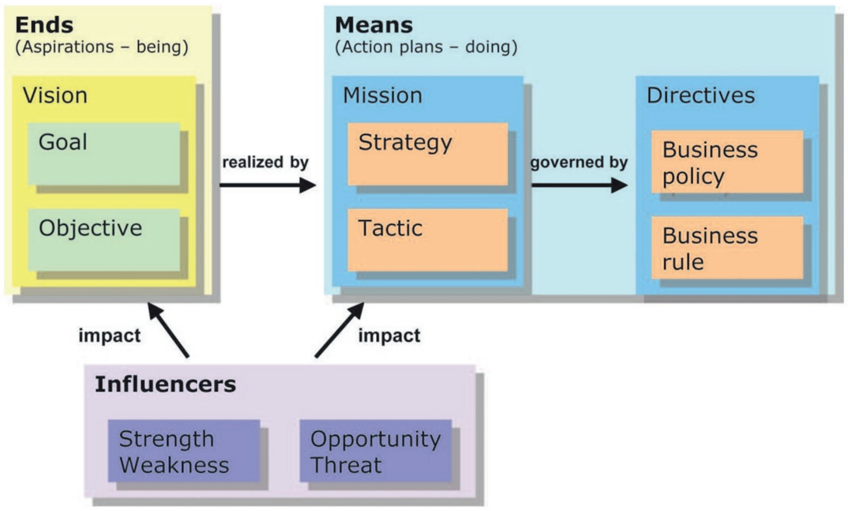


Figure 3.3: The Business Motivation Model.

3. Influencers, which describe the assessment of the elements that may influence the operation of the enterprise, and thus influence its ends and means.

3.3.3 i*

The i* framework [129, 128] focuses on concepts for modeling and analysis during the early requirements phase. It emphasizes the 'why' that underlie system requirements, rather than specifying 'what' the system should do. The i* framework has been developed to model and reason about organizational environments and their information systems. The central notion is the intentional actor.

Actors within an organization are viewed as having intentional properties such as goals, beliefs, abilities and commitments. Actors depend on each other to achieve goals, to perform tasks and to use resources. Furthermore, actors are strategic and will try to rearrange these dependencies to deal with opportunities and threats. Two types of models are distinguished: the SD model and the strategic rationale (SR) model. An SD model describes the dependencies among actors in an organizational context. A dependency models an agreement between two actors, where one actor (the depender) depends on another (the dependee) to fulfil a goal, perform a task or deliver a resource (the dependum). A dependency may involve a soft goal, which represents a vaguely defined goal with no clear criteria for its fulfillment. The SR model describes stakeholder interests and concerns, and how they can be addressed by various configurations of systems and environments. An SR model adds more detail to the SD model by looking 'inside' actors to model internal intentional relationships. Intentional elements, i.e. goals, tasks, resources and soft goals, appear both as external dependencies and as internal elements. Intentional elements can be linked by means-end relations and task decompositions. A third type of link is the contribution relation, which represents how well a goal or task contributes to a soft goal. The i* framework allows various types and levels of analysis, for example, to assess the ability, workability, viability and believability of goals and tasks.

3.3.4 KAOS

KAOS is a methodology for RE [23, 118, 119]. In comparison to i*, KAOS seems to focus more on the late requirements phase. Having said this, the goal concept in KAOS does allow one to model the motivations, i.e. the why, behind system requirements. But, in contrast to i*, KAOS seems less concerned with modeling the 'intentions' of actors.

The key concept underlying KAOS is goal. Van Lamsweerde [119] defines a goal as 'a prescriptive statement of intent that the system should satisfy through cooperation of its agents'. Here, an agent can be any actor involved in the satisfaction of the goal, e.g. an existing information system, an application to be developed, or a human user. Goals can be defined at different abstraction levels. Higher level goals and lower level goals are related through refinement relations, which define what lower-level goals are needed to satisfy a higher level goal. At the same time, these refinement relations define the justification for (why) a lower level goal is introduced. Typically, a (high-level) goal requires the cooperation of multiple systems. One important outcome of RE is the decision which goal can be automated (partly) and which not. A goal that is assigned to a system-to-be, such that the system is made responsible for the satisfaction of a goal, is called a requirement. Instead, a goal that is assigned to the environment of the system-to-be is called an expectation. Unlike requirements, expectations cannot be enforced by the system-to-be. In KAOS, a conflict relation can be used to model that the satisfaction of one goal prevents the satisfaction of another goal (and vice versa). An obstacle can be used to represent a situation that hinders or obstructs the satisfaction of some goal or requirement. An obstacle may be resolved by other goals. Further, KAOS allows the modeling of properties of the problem domain: domain hypotheses, which describe properties that are expected to hold, and domain invariants, which describe properties that always hold. KAOS supports various kinds of analysis, such as traceability, completeness, formal validation, refinement checking, and risk, threat and conflict analysis [119].

3.3.5 Observations

The following observations aim at guiding the decisions about the concepts that should be supported by ARMOR. The BMM cannot be considered a true requirements modeling language. The model focuses on business plans, which may involve high-level goals and objectives. A business plan that is developed using the BMM can be used as a starting point for (early-phase) RE. Elements of the BMM, such as goals and strategies, and also SWOT that result from the analysis of business influencers, may serve as sources or motivations for high-level goals. The i* framework focuses on the early requirements phase and is an expressive language, allowing various types of analysis. However, the expressiveness of the language and corresponding rich notation may be experienced as (too) complex and prevent people from using it [128]. Other observations are:

1. i* focuses on modeling the intentions of agents (actors) and allows the analysis of these intentions, concerning intentional concepts such as ability, workability, viability and believability;

2. the distinction between a means-end relationship and a decomposition relationship in i* in terms of semantics and consequence for further design steps is not always clear and may lead to confusion;

3. a similar remark can be made about the distinction between goals and tasks;

4. i* distinguishes between the internal intentions of an actor, and its external intentions in terms of dependencies on other actors. This is consistent with the distinction between the internal and external perspective on system design in ArchiMate.

The KAOS graphical notation [23] seems to be less complex and easier to use than i*. This comes at the price of less expressiveness, such as the inability to model the extent to which a goal contributes to another goal (although this ability can be introduced). Other observations are:

- KAOS does not use a separate actor model, but introduces the actors in the goal model via responsibility assignment relations;
- KAOS distinguishes between goals that typically must be satisfied by multiple cooperating agents, and requirements that are assigned to individual agents. This distinction corresponds to the distinction between activities and inter-activities (interactions, collaborations) in ArchiMate.

3.4 Language definition

In order to align the conceptual model of ARMOR with existing requirements modeling languages, the following approach is followed:

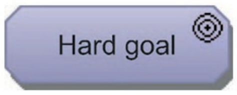
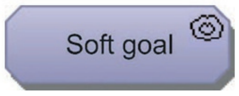
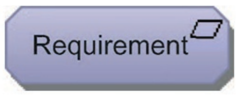
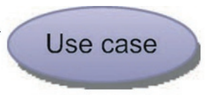
(1) Determine the common concepts underlying the languages studied in Section 3.3 and use these concepts as basis for ARMOR. This may involve the abstraction of concepts of one language to relate them to concepts of another language. (2) Extend the basic concepts of ARMOR in case its expressiveness is insufficient.

In these steps, guidelines like ‘KISS’ and suitability of the proposed concepts for the EA domain are taken into account. Furthermore, a ‘minimal’ set of generic concepts is strived for in order to keep ARMOR broadly applicable and to facilitate modifications and extensions later on when more experience has been gained with the use of ARMOR. The definition of ARMOR is divided into two domains: the requirements domain and the stakeholder domain.

3.4.1 Requirements domain

Table 3.1 depicts the requirements concepts that are supported by ARMOR, including their notation.

Table 3.1: Concepts of the requirements domain.

Concept	Notation	Concept	Notation
Hard goal		Soft goal	
Requirement		Use-Case	

Goals and Requirements

The key concept is the concept of goal, which is supported by BMM, i* and KAOS. Their definitions have in common that a goal represents some desired effect in the problem domain, or some desired properties of a solution. Furthermore, the goal concept can be used as an abstraction or generalization of other concepts:

- The concepts of vision and objective in BMM can be modeled as an abstract (high-level) and concrete (low-level) goal, respectively. Also the concepts of mission, strategy and tactic can, from a goal-oriented perspective, be seen as (sub-)goals that are obtained by 'operationalising' the concepts of vision, goal and objective, respectively.
- The concept of task in i* can be modeled as a concrete goal that defines how (part of) a more abstract goal can be satisfied.
- The goal concept in KAOS is an abstraction of the requirement and expectation concepts, since it abstracts from the agent (actor) to which the goal can be assigned.

An abstract notion of goal reduces the number of required concepts. However, this may be at the expense of precision and intuition. For example, a designer of a business plan does not only think in terms of 'goals', but specializes in terms of strategies, tactics, objectives, etc. For a similar reason, we want to distinguish between goals that can and cannot (yet) be assigned to actors. The distinction between hard and soft goals is made both in i* and KAOS (and implicitly in BMM via the distinction between goals and objectives). This distinction is considered significant and is therefore also supported in ARMOR. In particular, soft goals are useful in the evaluation of alternative designs. Based on the aforementioned analyses, we define the concepts of goal and requirement as follows:

- A goal models some end that a stakeholder wants to achieve. The desired end can be anything, e.g. some effect in or state of the problem domain, a produced value, tasks, or a realized system property. Hard goals are quantifiable goals with norms that specify when a goal is achieved. Soft goals are qualitative, i.e. not quantified, and in general more abstract. Typically, soft goals have to be refined into more concrete goals to make them quantifiable.
- A requirement models some end that must be realized by a single actor. A requirement can be considered as a specialization of a goal that is

delimited in scope and functionality, such that it can be assigned to a single actor.

- The modeling of (business) use cases is strongly related to the modeling of goals and requirements. Therefore, ARMOR also supports use-cases, similar to UML use-cases, including the include and extend relation. Use cases are used as a technique to elicit and specify system requirements. A use case describes the interactions between a system and some external actor, i.e. user [58]. This user typically initiates the use case having some goal in mind. This goal is satisfied when the use case completes successfully. Multiple, alternative sequences of interactions (called scenarios) may satisfy the goal. In addition, a use case may describe alternative sequences of interactions that handle failure, e.g. exception or error handling. By specifying only interactions, the system is considered as a 'and box', abstracting from internal detail. Because a use-case is associated with a single system that implements it, a use-case is considered as a type of requirement in ARMOR.

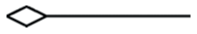
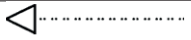
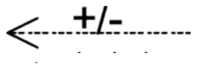
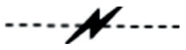
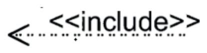
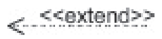
Relations in goal refinement

Table 3.2 depicts the relations that are supported with ARMOR, including their notations. BMM, i* and KAOS all support the refinement of goals into sub-goals. Moreover, BMM and i* distinguish two types of refinement relations: means-end relationships and decomposition relations. It is important to understand the distinction between means-end and decomposition. Existing languages like i* and KAOS do not distinguish correctly between these relations. Decomposition decomposes a goal into more concrete sub goals in such a way that a part-whole relationship exists between the goal and its decomposition. A means-end relation is much more a causal effect relation. The achievement of a means leads to the realization of the end. If there are multiple means identified, the realization of just one will facilitate the realization of the end. Therefore, we provide the following definitions:

- A goal decomposition decomposes an abstract goal into multiple more concrete sub-goals, such that the abstract goal is achieved if and only if all sub-goals are achieved. Typically, goal decomposition is used to define a goal more precisely, resulting in goal trees with measurable indicators, e.g. key performance indicators, at the leaves of the trees. A decomposition answers the WHAT question in goal refinement.

- A means-end relation relates a goal (the end) to some artifact (the

Table 3.2: Relations of the requirements domain.

Concept	Notation	Concept	Notation
Decomposition		Means-End	
Contribution		Conflict	
Include		Extend	

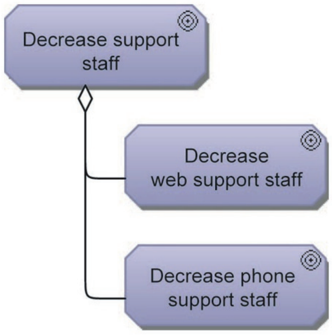


Figure 3.4: Example goal decomposition

means) that realizes the goal. This artifact can be another goal or requirement, or can be an architectural element, such a service, process or application. While goal decomposition is used for the more precise definition of goals, the means-end relation is typically used to illustrate causal effect relations between goals and other artifacts. The means-end relation answers the HOW and WHY questions in goal refinement.

Figure 3.4 illustrates a decomposition. It creates a definition of what decreasing the support staff actually means. In this example the goal decrease support staff is only achieved if both the web support staff and phone support staff are achieved. The stakeholders create a definition of what should be done in more detail.

Figure 3.5 illustrates a means-end relation. Reducing the amount of customer calls is believed to realize decreasing the support staff. By reducing the amount of customer calls we can reduce the support staff. The stakeholders believe there is a cause effect relation between these two goals. This causal

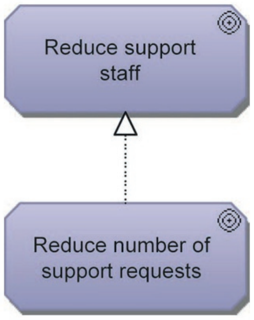


Figure 3.5: Example means-end relation

effect is based on some goal theory, or assumptions, from the stakeholder that if customers stop calling, you do not need staff answering them.

Conflicts, obstacles and contributions

Both i* and KAOS allow one to model that some goal or situation has a negative influence on the satisfaction of another goal.

- KAOS supports the conflict relation and the obstruct relation in combination with the obstacle concept. Furthermore, the resolution relation can be used in KAOS to resolve, i.e. ‘dissatisfy’, an obstacle;
- i* supports the contribution relation to model positive and negative influences on the satisfaction of soft goals. These influences are defined in qualitative terms, e.g. using the range: ++ + +- - –. The obstruction of goals by obstacles is not modeled as part of a goal model in ARMOR. An obstacle is considered the result of the assessment of some stakeholder concern, like the assessment of an influencer as a threat or weakness in the BMM. The modeling of assessments should however be supported by ARMOR – not as part of the goal domain – but as part of the stakeholder domain.

The following relations can be modeled as part of goal models in ARMOR:

- A contribution relation from some goal G1 to another goal G2 to represent that G1 contributes (influences) the satisfaction of G2 positively or negatively. Typically, the contribution relation is used to facilitate the evaluation of alternative goal refinements. The need to be able to qualify the strength of the contribution, and in what detail may depend on the situation at hand. Therefore, different qualification ranges may be used.

- A conflict relation between two goals G1 and G2, such that the satisfaction of G1 inhibits the satisfaction of G2, and vice versa. A conflict is only possible between hard goals (and requirements), since the criteria for the satisfaction of soft goals is unclear; i.e. it is unclear when the satisfaction of a soft goal inhibits the satisfaction of another goal.

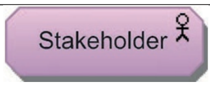
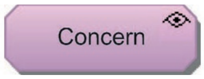
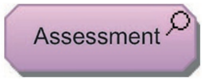
Assumptions

The refinement of some goal may be based on certain assumptions about (elements in) the problem domain. i* and KAOS introduce the notions of assumption, belief and domain property for this purpose. Since it is considered useful to make such assumptions explicit, ARMOR supports the general notion of 'assumption', however an assumption is not recorded as a concept but as a property of a goal. Our main design goal for ARMOR is to keep it as simple as possible. We chose not to model an assumption as a construct but as a property of a goal. This is seen as contextual information of a goal. We wish to be able to record this contextual information, but not introduce an additional modeling concept.

Stakeholder Domain

Table 3.3 depicts the concepts of the stakeholder domain, which are used to model the origin and owners of goals and requirements. The concepts have the following interpretation:

Table 3.3: Concepts in the stakeholder domain

Concept	Notation
Stakeholder	
Concern	
Assessment	

- A stakeholder represents an individual, team, or organization (or classes thereof) with interests in, or concerns relative to, the outcome of the architecture. This definition is adopted from TOGAF [115]. A stakeholder typically associates value to certain aspects of the enterprise, and thus also its reflection in the enterprise's architecture. Examples of stakeholders are not only the board of directors, shareholders, customers, business and application architects, but also legislative authorities.

- A concern represents some key interest that is crucially important to certain stakeholders in a system, and determines the acceptability of the system. A concern may pertain to any aspect of the system's functioning, development or operation, including considerations such as performance, reliability, security, distribution, and evolvability. This definition is also adopted from TOGAF[115].

- An assessment represents the outcome of the analysis of some concern. This outcome may trigger a change to the EA, which is addressed by the definition of new or adapted business goals.

The association relation of ArchiMate is (re-)used to relate stakeholders to concerns and concerns to assessments. A stakeholder can have one or more concerns, and a concern may be shared by multiple stakeholders. An assessment typically assesses a single concern, but could involve multiple concerns. A concern may be analyzed through different assessments.

3.4.2 Meta Model

Figure 3.7 depicts the abstract syntax, or meta-model, of ARMOR. The colored classes represent the new concepts introduced by ARMOR, the white classes represent the reused concepts from ArchiMate. Concepts from the stakeholder domain are: stakeholder, concern, assessment and their relations: stakeholder concern relation, concern assessment relation and assessment goal relation. These relations are all based upon the existing association relation from ArchiMate. In the requirements domain we find the goal concept, which is specialized into hard goal and soft goal. The hard goal is further specialized into requirement. A use case is a specialization of a requirement. The contribution and conflict relation are two new relations. The means- end relation is a specialization of the ArchiMate realization relation. The goal decomposition relation is a specialization of the ArchiMate aggregation relation.

The realization of requirements through ArchiMate concepts corresponds with the way KAOS relates requirements to behavior, actors and data. Behavioral elements realize the requirements; actors are responsible for performing this behavior and data is processed through performing this behavior, see Figure 3.6. In this figure an insurance service is used to realize a functional requirement sell insurances over the internet. The service is assigned to an actor insurance department as behavior is performed and some business information is involved.

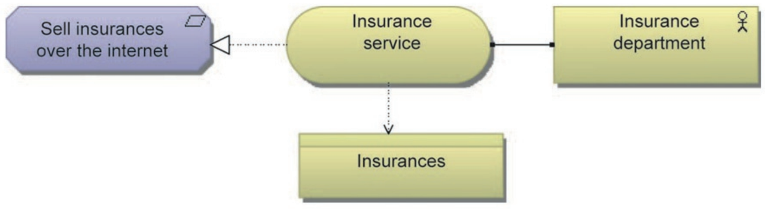


Figure 3.6: Realizing requirements by architecture elements.

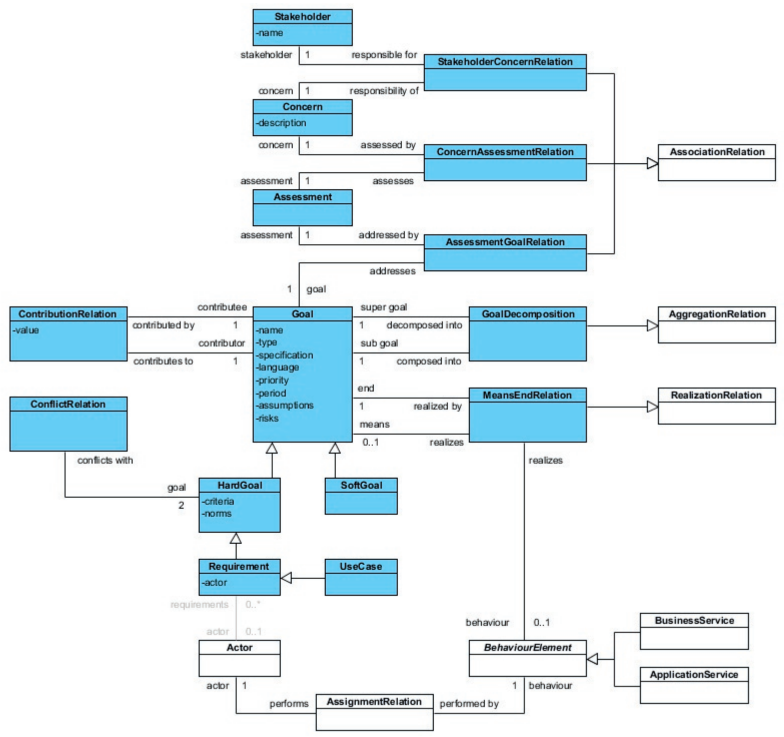


Figure 3.7: The meta-model of ARMOR.

3.5 Application of ARMOR

We will demonstrate the application of ARMOR through a real life example in the healthcare industry. Through the remainder of this example we will call the hospital where this project took place Hospital X. This particular hospital specializes in movement and posture disorders.

3.5.1 Case Description

Hospital X strives to ensure satisfied patients, responsible and satisfied employees, safe and efficient care, steering based on quality and service and finally accomplishment and growth. Hospital X elaborated this in its mission, which is included in the policy plan; this mission has four core elements:

- Patient focus, deliver care based on the needs and wishes of the patient;
- Excellence, develop treatments based upon the latest scientific breakthroughs;
- Innovation through research and development;
- Entrepreneurial through expanding and seizing business opportunities.

Hospital X experienced a number of problems, the major concern being that patients had trouble with the lengthy registration process, which was basically in conflict with their patient focus goals. Secondary concerns were incomplete patient information, incomplete information delivery and no insight in whether patients should be billed or not based upon their insurance status. Last but not least, the new electronic patient record (EPR) act forces hospitals to record patient information in a national database to improve information delivery between care givers. Hospital X also identified a number of business opportunities, healthcare 2.0 being the most important one. Healthcare 2.0 is patient centered working using the most modern (communication) technologies available.

The project goal was to elicit high-level business requirements and analyze these using the modeling language presented in this chapter. We used two workshops to elicit and validate the high-level requirements and designed a to-be architecture based on these results. This part of the architecture was then discussed and approved by the senior information architect at the hospital.

Through this example we will demonstrate that ARMOR facilitates EA design through relating stakeholder goals to elements from the EA through goal refinement, thus demonstrating the need for that particular architecture.

We will demonstrate that: (i) ARMOR captures the relevant business context, comprising assessments of stakeholder concerns and information found in business plans; thus facilitating improved traceability from high-level goals to architecture requirements and architecture elements, (ii) that this traceability facilitates adapting to change easier, through following the relationships from high-level strategic goals to the architecture realizing it, (iii) ARMOR also facilitates improved detection of conflicting interests and solutions and (iv) using ARMOR it becomes possible to evaluate alternatives against soft goals from the organization.

3.5.2 Goal Modeling with views

This example uses three kinds of modeling views. The first view is the stakeholder view. In this view we model the relevant stakeholders of this project, their concerns, assessments of these concerns and the first high-level goals. The second view we use is a goal refinement view. Here we select a stakeholder concern and refine the initial goals into requirements for the EA. The final view is a requirements implementation view. Here we show which architectural elements realize the requirements. This example is structured according to these views. We will provide at least on example of each view.

3.5.3 Stakeholder View

This stakeholder view is limited to the board of Hospital X, as shown in Figure 3.8. The board is concerned with innovation, patient focus, excellence and entrepreneur- ship. These concerns were identified in the policy plan of Hospital X. Two major assessments were identified at Hospital X. To address the innovation concern all new products and services should fit within healthcare 2.0.

Healthcare 2.0 means patient centered working and using modern technology to bring the healthcare provider and the patient closer together and give the patient a sense that he is more involved in the care process. A second area of concern is the current patient registration system. This system is troublesome for the patient; he has to wait in line before he can register himself as a patient at the hospital. Patient registration is a lengthy process, and has a negative impact on the patient, where he is preoccupied with his illness and should not be bothered with a lengthy registration process.

Based upon these assessments and concerns Hospital X identified a number of change goals, patient registration should happen based upon consumer feedback, healthcare 2.0 has to be introduced in the organization and new products

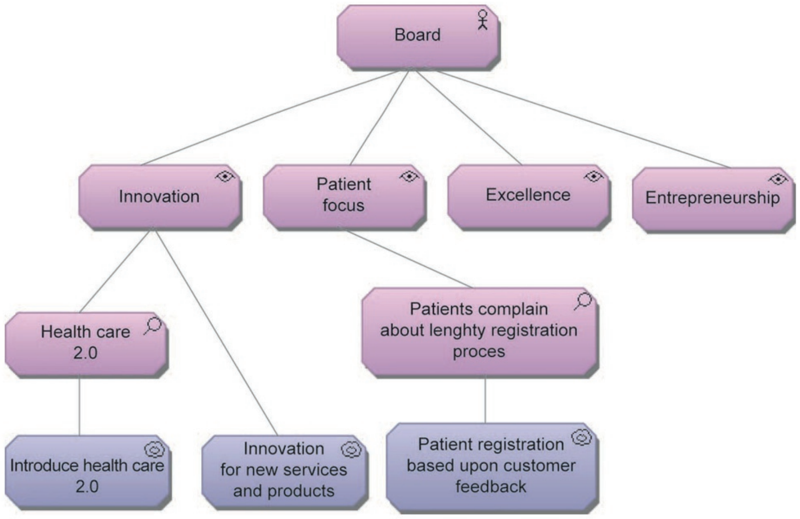

Board
Innovation
Patient focus
Excellence
Entrepreneurship
Health care 2.0
Patients complain about lenghty registration proces
Introduce health care 2.0
Innovation for new services and products
Patient registration based upon customer feedback

Figure 3.8: An example goal model based on the stakeholder view.

and services should be developed on an innovative way. Based upon surveying the customers they decided to improve the patient registration process.

3.5.4 Goal Refinement View

Figure 3.9 models the goal refinement view. In this view the initial goals are structured and refined. To demonstrate the goal refinement view we refine the security concern from the information manager. His main goal was to provide a safe to use service. Safety is by itself a vague statement, therefore we modeled this as a soft goal, it requires a decomposition into measurable goals. Safety in this case is decomposed into 'protect identity of the patient' and 'protect patient information'. Protecting the identity of the patient is realized through checking the identity of each individual patient. This is decomposed into automated and manual checks. Protecting personal information is realized through denying access to third parties and this led to the requirement of encrypting the patient information.

3.5.5 Requirements Realization View

Figure 3.10 depicts a so-called requirements implementation model. Requirements are realized by some form of behavior from the EA, which falls under the responsibility of an actor and some data is processed. In this view we operationalize the requirement for manual checks through a business process 'patient identification', which is the responsibility of the employee at the registration desk and during this process identification information is used.

3.6 Tool Support

Any modeling language, especially when applied in realistic situations where models may become quite large, can only be successful if supported by adequate and professional tooling. A good model editor makes sure that the language is applied correctly and consistently, and may offer facilities for, among others, version control and multi-user editing. A modeling tool may also support the use of viewpoints and views on models. We have implemented the ARMOR language in the architecture modeling tool BiZZdesign Architect (see http://www.BiZZdesign.nl), as an add-on to the ArchiMate language. This tool provides all of the functionalities described above. Visualization and analysis of models can hardly be carried out by hand and requires tools as well. For ARMOR, we have implemented a number of useful analysis techniques for requirements models in BiZZdesign Architect, which we will briefly describe below (illustrated with the Hospital X example).

3.6.1 Traceability

Figure 3.9 depicts a viewpoint based upon a stakeholder concern. What we see here is that the board is concerned with innovation. An assessment of this concern reveals that using healthcare 2.0 is an opportunity in new product and service development. Two goals are elicited for this concern and these goals are realized through the first high-level requirement that Hospital X should offer some sort on online registration from home. Figure 3.10 depicts the first architectural model based upon this requirement. In this way, we can trace from the stakeholders, their concerns and assessments from these concerns to the high level goals, refined goals, requirements and ultimately the relevant enterprise architectural elements.

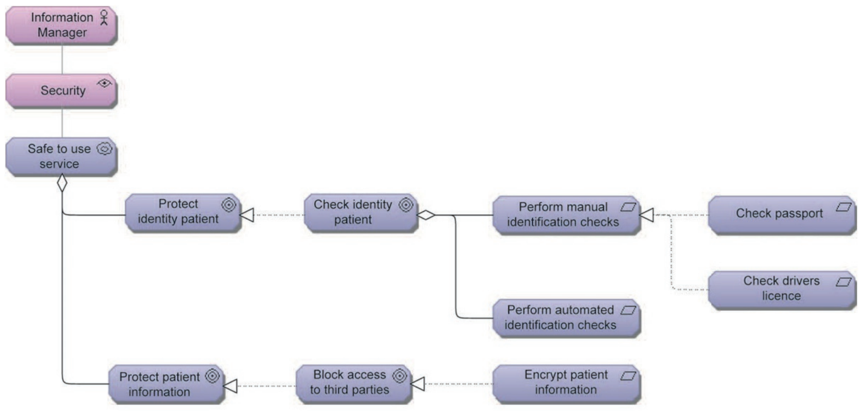

Information Manager
Security
Safe to use service
Protect identity patient
Check identity patient
Perform manual identification checks
Check passport
Check drivers licence
Perform automated identification checks
Protect patient information
Block access to third parties
Encrypt patient information

Figure 3.9: A more elaborate goal model using the goal refinement view.

3.6.2 Impact of Change Analysis

Traceability of stakeholder concerns introduces powerful analysis possibilities; one of these is the impact of change analysis. To demonstrate this analysis we use the goal models from figures 3.9 and 3.10. This impact of change analysis is one of the more powerful analysis types. Every organization interacts with the environment; this environment contributes to the goals of the organization. One of the high-level goals here is to comply with the data protection act, which is decomposed into the requirement check identity of the patient and compliance with security guidelines. Suppose changes occur in the legislation concerning the data protection act, through these goal refinement links we can trace into the architectural models and analyze which parts of the organization are affected by these changes. This way we know which services processes, actors and IT systems are affected by a change.

3.6.3 Detection of conflicts

We were able to identify conflicting goals between two different requirements views. The manager care was interested in creating an information exchange service based upon registered patient data. However, this was not allowed because patient data are confidential and this was contained in the information security guidelines within the hospital, see figure 3.11. Because of the importance of complying with information security guidelines, extracting patient

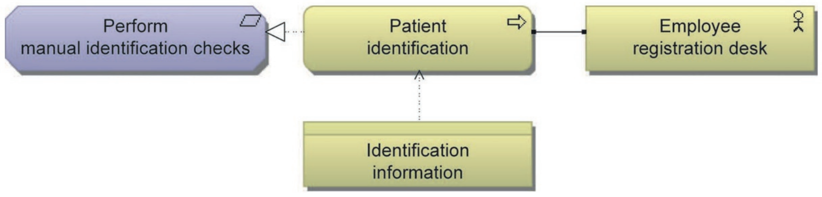

Perform manual identification checks
Patient identification
Employee registration desk
Identification information

Figure 3.10: Goal realization by an element from the Enterprise Architecture.

information from registration data was dropped as a goal by the manager care.

3.6.4 Evaluation of Alternatives

Two main solution alternatives emerged at Hospital X. One possibility was that customers preregister from home via the internet. The second alternative was that they could register at different sign-in stations located throughout the hospital. The latter is also used by Schiphol airport for a speedy check-in procedure. Based upon these alternatives Hospital X selected a number of important soft goals. These alternatives are then evaluated based upon their contribution to said soft goals. Figure 3.12 the modeled results from this evaluation. The presented alternatives were evaluated against a number of soft goals. In this case stakeholders 'graded' the contributions on friendliness, safety, improved registration, feedback from customers and healthcare 2.0. Based upon these contribution models Hospital X decided to pursue the online registration service.

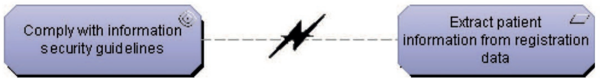

Comply with information security guidelines
Extract patient information from registration data

Figure 3.11: Conflicting goals, illustrated by the conflict relation.

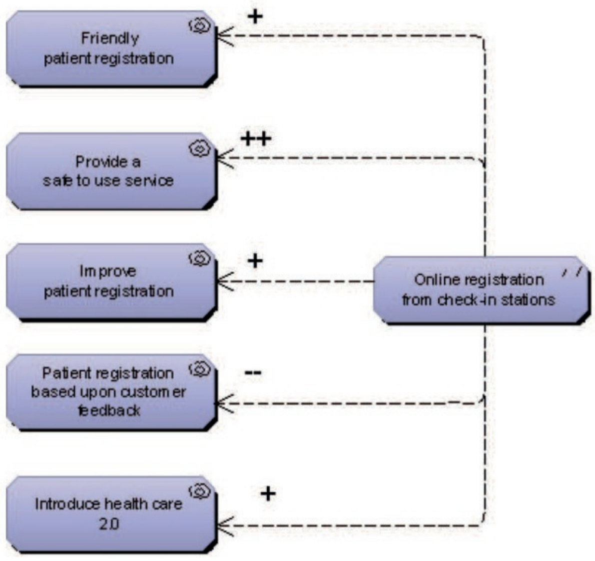

Friendly patient registration
+
Provide a safe to use service
++
Improve patient registration
+
Online registration from check-in stations
Patient registration based upon customer feedback
--
Introduce health care 2.0
+

Figure 3.12: Alternative evaluation using the contribution relation for the online registration from check-in stations.

3.7 Outlook Architectural Principles

An important research development in the field of EA is the use of principles [76, 92, 11]. This section will describe the future development of ARMOR to integrate principles into the language.

A precise definition of the concept of principles as well as the mechanisms and procedures needed to turn them into an effective regulatory means still lack [108]. This article will only discuss the generic attributes of a principle and how principles can supplement the ARMOR language. Generally speaking, two different types of principles can be distinguished, *normative principles* and *engineering principles*. Normative principles are rules of conduct, or guidelines. They provide a norm that designers have to take into account. A normative principle can be broken; someone can choose not to follow them.

Engineering principles describe some law of nature that underlies the working of some artificial device. Engineering principles are based upon causality, if we apply this principle these effects will happen. An example of an engineering principle is that any object, wholly or partially immersed in a fluid, is buoyed up by a force equal to the weight of the fluid displaced by the object.

An engineering principle is more a fact and cannot be broken. Architectural principles are normative principles based upon causality motivating the working of this principle. In more concrete words, architectural principles use the business drivers as the goal they should realize, for example business goals, architecture goals and IT goals. When an architectural principle is applied, the organization expects that it has some positive contribution on the business drivers, this is the causality aspect. This is very similar how we use the contribution relation in ARMOR.

For example, when an organization identifies that user productivity is a goal they should pursue, they can identify principles that realize this goal. 'All applications must be easy to use' could be such principle. It is based on the assumption (or fact) that when users do not have to struggle with learning the applications they are more productive.

Since an architectural principle is a normative principle it *constrains* the solution space of the designer. A principle applies to *all solutions* in a *particular context* and should be stable enough to be reusable.

This chapter discusses the modeling of goals and requirements for organizations. A goal is a desired state or desired effect that a stakeholder wants to bring about in the problem domain. A requirement is also a desired state or desired effect, but for an actor instead of a stakeholder.

In this context a principle is a *preferred way* to reach a desired state. A principle brings about a desired proven (or assumed proven) effect in the problem domain, by constraining the solution space.

A principle is not a requirement. A requirement is associated with a *single solution,* whereas a principle is associated with *all solutions in a particular context*. When a principle has to be applied to a single solution it needs to be refined into requirements. This is where the introduced goal refinement techniques from this chapter can assist.

For example, the principle 'compliance with law' is provided by TOGAF as an architectural principle. Applying this principle leads to the desired effect that the legislators will not sanction the organization. This principle constrains every possible solution an organization can develop, from business services to IT infrastructure. However, it is still too general to be applied to a single solution. It therefore requires refinement, much in the way this chapter proposed. Compliance with law could be decomposed into more concrete legislation that applies to the organization and then refined into requirements.We expect to adapt ARMOR in the future with the principle concept.

3.8 Conclusions

In this chapter, we have presented a language, called ARMOR, for modeling goals and requirements in EAs. The origin of high-level goals is modeled in terms of stakeholders, their concerns and the (SWOT) assessments that are addressed by the goals. Goals are refined into (alternative sets of) sub-goals, via goal trees. Low-level goals (requirements) are related to the services, processes and applications that implement the requirements. This enables forward and backward traceability between goals and requirements.

The ARMOR language is based on the existing requirements modeling languages and is aligned with the standard enterprise modeling language ArchiMate. This brings existing theory and analysis techniques to the domain of EA modeling. We have demonstrated how to realize traceability of stakeholder concerns to the architectural elements. This traceability is realized through goal refinement and providing means to integrate this into the architecture domain. We are able to model and refine strategic goals and policies found in business plans.

Through goal refinement we are able to link this business context to the new architecture elements, thus realizing traceability. Through capturing these links it becomes possible to reason about the effects of changing goals on the EA. Through following links from goals via requirements to EA elements, we can derive which architectural elements are affected by a change in a high-level goal. We also showed how ARMOR can be used to support stakeholders in reasoning about conflicting interests and solutions. Visualizing the effects of conflicting goals helps to understand what the effects are of these conflicts on the EA and leads to dropping or changing certain goals by certain stakeholders. Finally we presented the use of ARMOR to reason about two emerging solution alternatives and to evaluate them based upon the soft goals provided by the stakeholders. Through explicitly modeling these contributions the stakeholders can select the most favorable alternative.

Currently, we apply ARMOR combined with an architecture-oriented RE approach in a number of consultancy projects. These projects help to validate and improve ARMOR and the associated approach. For example, through applying ARMOR in practice we evaluate the appropriateness and completeness of our concepts and derive guidelines for goal decomposition and means-end analysis. These can be captured in consistency rules and refinement patterns. Consistency rules guarantee the correctness of the requirements model and refinement patterns describe common goal refinements.

4

First Evaluation[1]

[1]This chapter is based on an article in the proceedings of the International Working Conference on Requirements Engineering: Foundation for Software Quality [36]

4.1 Introduction

In large companies the gap between business and IT is usually bridged by designing and maintaining a so-called *enterprise architecture* (EA), which is a high-level representation of the enterprise, used for managing the relation between business and IT. A full-scale EA consists of (i) an architecture of the business, in terms of products, services and processes, (ii) an application architecture in terms of of application components, functions and services, (iii) an infrastructure architecture in terms of servers, mainframes, network, and (iv) the relationships between these different architectures [115].

Enterprise architectures are typically modeled in larger organizations (say starting from 500 employees) and are used to coordinate IT projects and to manage the cost of IT. Increasingly, they are also used to increase flexibility of the organization and to justify the contribution of IT to business goals. This requires traceability of business goals to IT architecture (to quickly identify the impact on IT of changes in business goals) and of IT architecture to business goals (to justify the contribution of an IT component to a business goal). This requires a goal-oriented addition to the current crop of EA modeling languages.

An important constraint is that we want the resulting language to be usable and useful for enterprise architects in practice. Usability means at least tool support and understandability for the architects; utility means that the resulting language and tool can indeed be used to realize traceability in practical cases. In this chapter, we evaluate the language defined in chapter 3 in terms of understandability and utility. Section 4.2 discusses the research methodology. Section 4.3 summarizes the definition of ARMOR. Section 4.4 introduces the first case study and section 4.5 introduces an initial redesign. This redesign is evaluated in section 4.6. We end this chapter with a discussion of the results in section 4.7.

4.2 Research Methodology

We used a design research methodology in which we alternate over an engineering cycle, where we design an artifact, and a research cycle, where we investigate the properties of this artifact and of the problems it is intended to solve [127]. Figure 4.1 shows that we executed the engineering cycle twice. In the first iteration, we investigated the problem to be solved, designed a method called ARMOR to treat the problem (section 4.3), supported by a tool for editing and traceability analysis[2] and validated the artifact (section 4.4).

[2]`http://www.bizzdesign.nl/download/downloads-trial-software`

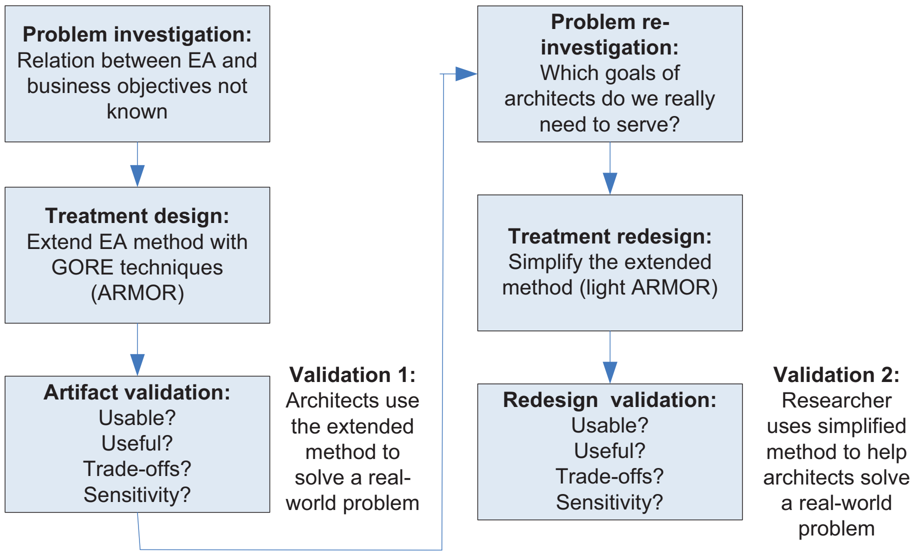

Problem investigation:
Relation between EA and business objectives not known
Treatment design:
Extend EA method with GORE techniques (ARMOR)
Artifact validation:
Usable?
Useful?
Trade-offs?
Sensitivity?
Validation 1:
Architects use the extended method to solve a real-world problem
Problem re-investigation:
Which goals of architects do we really need to serve?
Treatment redesign:
Simplify the extended method (light ARMOR)
Redesign validation:
Usable?
Useful?
Trade-offs?
Sensitivity?
Validation 2:
Researcher uses simplified method to help architects solve a real-world problem

Figure 4.1: Design research methodology of this chapter

In the second iteration, we stripped ARMOR to its essentials, called Light ARMOR (section 4.5), and validated this lightweight version and supporting tool (section 4.6).

ARMOR is an extension of an EA modeling language called ArchiMate 1.0 [73] with goal-oriented requirements engineering (GORE) techniques [29, 101]. We call this a *treatment* rather than a solution because it would be simplistic to assume that any real-world problem can be totally solved, just as it would be simplistic to assume that any medical problem could be totally eliminated by a medicine.

ARMOR combined concepts from all well-known GORE languages, which is why this research also provides insights into GORE concepts in general. To validate ARMOR, we taught the method to enterprise architects of a large governmental organization, who then used it to perform an EA design project. This is a form of *technical action research* (TAR), in which an artifact is validated by actually using it to solve a real-world problem. This TAR project itself has the structure of an engineering cycle performed by the enterprise architects (figure 4.2).

These insights from case study 1 led to an improved problem understanding and in a second engineering cycle we simplified ARMOR in the light of the

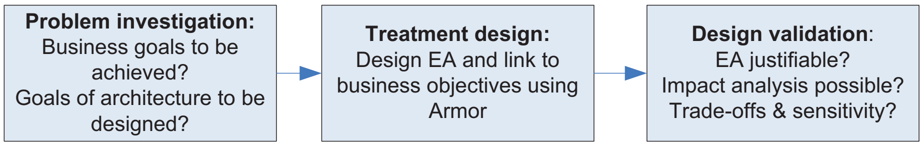

Problem investigation:
Business goals to be achieved?
Goals of architecture to be designed?
Treatment design:
Design EA and link to business objectives using Armor
Design validation:
EA justifiable?
Impact analysis possible?
Trade-offs & sensitivity?

Figure 4.2: Structure of validations 1 and 2

lessons learned. Light ARMOR was then used by to design an EA for another client, acting as consultant. This is validation 2 in figure 4.1. This is a second TAR project, but this time with the researcher as actor, rather than the client itself, as in validation 1.

The lessons learned from validation 2 were used to answer the researchers' validation questions about Light ARMOR. These answers were then generalized to GORE concepts in general, when used in similar contexts (section 4.7).

4.3 Definition of ARMOR

Table 4.1 lists the major GORE concepts and shows how we have used them in ARMOR. The following list summarizes the motivation for the construction of ARMOR. More detail is provided elsewhere in chapter 3 and the original articles [29, 101].

- Goals belong to *stakeholders,* and different stakeholders may have conflicting goals. This is important in practice but is left undefined in most GORE languages, although the i* concept of intentional actor has some similarity with our stakeholder concept. We have adopted the stakeholder concept of TOGAF [115].

- BMM, i*, and KAOS all define a *goal* as an end (or desire or intention) of a stakeholder but differ in defining this goal as a property of the system or of its environment. We define goal as some end a stakeholder desires to achieve and leave open what it is a property of.

- We follow i* in distinguishing *hard* and *soft* goals but make the requirement "clear satisfaction criteria" explicit by requiring measurability.

- Goal *decomposition* is in terms of conjunction of sub goals. It is called “refinement” in KAOS. Tropos uses the concept of satisficing. i* and BMM have rather vague definitions.

Table 4.1: Overview of GORE and ARMOR constructs

GORE construct	**ARMOR construct**
“Organizational actors are viewed as having intentional properties such as goals, beliefs, abilities, and commitments” i* [129].	A *stakeholder* is an individual, team, or organization (or classes thereof) with interests in, or concerns relative to, the outcome of the architecture ARMOR [29]. adopted from TOGAF [115].
”Goals are desired system properties that have been expressed by some stakeholder(s)” KAOS [119]. ‘Goals are the intentions of a stakeholder” i* [129].	A *goal* is some end that a stakeholder wants to achieve [29].
“Hard Goals are the intentions of a stakeholder” i* [129].	A *hard goal* is a goal with measurable indicators [29].
“Soft Goals are goals without clear satisfaction criteria” i* [129].	A *soft goal* is a goal without measurable indicators [29].
“An element that is linked to its component nodes” i* [129]. “An end that includes an other end” BMM [16]. ”The parent is satisfied if all of the offspring are satisfied” Tropos [13]. “The conjunction of all the sub goals must be a sufficient condition entailing the goal” KAOS [119].	A goal can be *decomposed* into two or more concrete sub-goals, such that the goal is achieved if and only if all its sub-goals are achieved.
“The contribution of a design on a qualitative goal ...” KAOS [119]. “Link elements to a soft goal to analyze its contribution” i* [129]. “Contribution analysis identifies goals that can contribute positively or negatively in the fulfillment of the goal to be analyzed...” Tropos [13].	A goal G1 *contributes* to another goal G2 if satisfaction of G1 influences the satisfaction of G2 positively or negatively [29].
“These links indicate a relationship between an end, and a means for attaining it i* [129]”. ”Relationship linking a requirement to operations KAOS [119]”.	A *means-end* relation relates a goal (the end) to some artifact (the means) that realizes the goal [29].
“Goals are conflicting if under some boundary condition the goals cannot be achieved altogether” KAOS [119]”.	A *conflict* relation exists between two goals if under some boundary conditions they cannot be achieved together [29].
“Goal assigned to an agent of the software being studied. KAOS [119]”. “A quantitative statement of business need that must be met by a particular architecture or work package” TOGAF [115] .	A *requirement* is some end that must be realized by a single component of the architecture [29].
“Concerns are the key interests that are crucially important to the stakeholders in the system, and determine the acceptability of the system” TOGAF [115].	A *concern* is some key interest that is crucially important to certain stakeholders in a system, and determines the acceptability of the system [29].
“An Assessment is a judgment about some Influencer that affects the organization’s ability to employ its Means or achieve its Ends BMM [16]”.	An *assessment* is the outcome of the analysis of some concern [29].

- The *contribution* relation is defined most clearly in Tropos and is taken to mean influence, positive or negative.
- The *means-end* relation is used in i* to identify tasks to realize goals and in KAOS to identify operations to realize goals. In ARMOR we define it as relating a goal (the end) to some artifact (the means) that realizes the goal. This artifact can be anything, such as a goal, requirement or an element from the architecture.
- Only KAOS defines the *conflict* relation. However we believe it to be so different from the contribution relation that we include it, adopting the KAOS definition.
- KAOS is also the only GORE language that explicitly defines the *requirement* concept. It is defined as a concrete goal that has been assigned to a single actor. TOGAF defines requirement as a business need allocated to an architecture. The ARMOR definition combines these two definitions.
- The concepts of *concern* and *assessment* are not part of GORE but of the EA literature. We therefore included these concepts, taking our clues from BMM and TOGAF.

ARMOR has a notation that extends the EA language ArchiMate 1.0 [73], and tool support in the form of an editor. The editor supports the creation of integrated goal models and EA models. The tool also provides functionality to trace requirements to EA and vice versa. The resulting language served as an input for the ArchiMate 2.0 specification process. The notation is described and motivated elsewhere [29, 101] and does not concern us here.

Figure 4.3 shows the core part of ARMOR's meta model. Cardinalities are not shown so as not to clutter up the diagram, except the cardinality from requirement to architecture component, which is many-one. The diagram shows that stakeholders have concerns, that they assess in a certain way, which leads to goals, that are hard or soft; hard goals can be requirements, and each requirement is allocated to exactly one architecture component. Goals can be decomposed, can have contribution and means-end relations, and they can conflict. The complete meta-model of ARMOR has been described elsewhere [29].

4.4 Case Study 1

To validate ARMOR we first wanted to test usability by enterprise architects. The further question of utility can only be answered once we have a usable

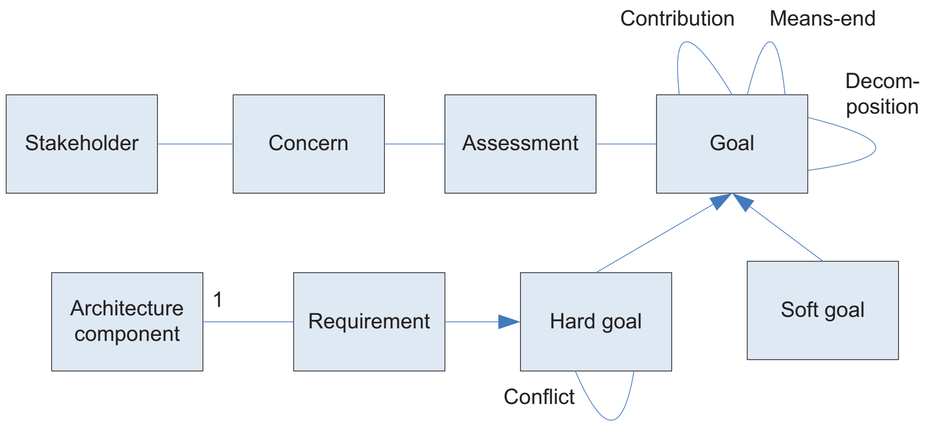


Figure 4.3: ARMOR's meta-model The arrow represents specialization. Cardinalities are not shown in the figure.

language. However, we did want to know whether ARMOR misses potentially useful constructs. We therefore identified the following research questions.

- Q1. What constructs of ARMOR do enterprise architects use in practice?
- Q2. Why (for which purpose) do they use these concepts and relations?
- Q3. Is this the intended use of the constructs?
- Q4. Which construct not in ARMOR are considered by architects useful additions to ARMOR?

The only way to answer these questions is to have practicing enterprise architects use ARMOR and observe how they do it. Since ARMOR will not be transferred to a practical context unless we do the transfer, we needed to perform an action case study, where we first transferred knowledge of ARMOR to a company and then observed ARMOR use.

4.4.1 Case description and research design

The case study took place at a large governmental organization in the Netherlands that we will call Organization 1. The organization is responsible for state pensions and child support payments by the Dutch Government. The budget available for these payments is around thirty billion euros, consisting entirely of taxpayer money. The company employs around 3000 civil servants distributed over several locations in the country. Relevant stakeholders include enterprise architects and information analysts, who are looking for a technique that can show the value of their designs to business stakeholders. Relevant stakeholders also include information managers, who are looking for a technique that would enable them to analyze the effect of changing organization goals on the EA.

Organization 1 contacted BiZZdesign if they could help with improving traceability between the business objectives and the enterprise-architecture. BiZZdesign offered to provide ARMOR with tool support, which the organization accepted.

We provided a one-day training on ARMOR to six enterprise architects of Organization 1. The architects of Organization 1 then proceeded to create ARMOR models of business goals and their links to the existing EA. They did this on their own, by investigating business documents of Organization 1 and by conducting workshops. No help was provided. However, we visited Organization 1 every two weeks to review the models made by the architects and to provide advice. On those occasions we also made notes of discussions among the architects.

To summarize, the treatment applied to the case consisted of (1) a one-day training and (2) bi-weekly advice. Data collection took place by collecting documents produced by the architects and by making notes during discussions among architects. There was no possibility to collect observations by other means, such as questionnaires or interviews, as the enterprise architects were too busy for that.

4.4.2 Observations and explanations

We extracted the following observations from the data.

- The architects used the *stakeholder* concept as intended, to record the existence of some entity that has a stake in the development of the organization. The (obvious) explanation is that the stakeholder concept is widely known in businesses, and has a meaning well-captured by the TOGAF definition that we adopted.

- The architects also used the *goal* concept as intended. This too is a concept well-known in the practice and theory of business management. However, they did not see why the distinction between soft goals and hard goals would be relevant in their models. This is explained by their way of working: The architects started out identifying relevant business goals and then proceeded, later on in their work, to decompose these into key performance indicators (KPIs). So initially, all goals are soft; eventually, all goals are decomposed into hard goals. For example, the soft goal to maintain quality of service was decomposed into the goals to maintain timeliness of service requests and to maintain legality of service, which are hard goals because measurement procedures were defined for them: the maximum amount of time for a service request, and for every decision a reference to the law on which the decision is based, must be documented. They did not see the point of making this transition explicit by using a different symbol for soft and hard goals.

- The *decomposition* relation was used as intended: to refine a goal into more concrete sub-goals, in such a way that achievement of the conjunction of the sub-goals implies the achievement of the higher level goal. For example, the goal to decrease cost was decomposed into the sub-goals to decrease cost of internal services, to decrease cost of external services and to decrease cost of IT.

- The *contribution* relation was used by the architects as intended, namely to indicate that achievement of one goal influences the achievement of another goal. For example, the goal to increase automatic service delivery contributed positively to the goal of decreasing cost of external services.

- The *means-end* relation is constrained in the ARMOR tool to be an influence relation from a system requirement to a goal. This was understood by the architects and they used it in this way. But they did not understand why a separate means-end relation was included to represent this, where a contribution relation expresses in their view exactly the same thing: Influence.

- The *conflict* relation was not used by the architects in this case. The architects explained that in this case there simply were no conflicts between different stakeholder goals. In addition, they did not see any difference between a conflict and a negative contribution.

- In ARMOR, a *requirement* is a goal that must be achieved by a single component of the architecture. This definition was not quite understood by the architects, and they often formulated requirements that were not goals of a single architecture component. An example of this is the “requirement” that the use of marketing techniques must be improved. This is a business goal, not a system requirement.

- The architects had difficulty understanding the difference between *concerns* and goals. The intention of the concept is that it be used for areas of concern for the stakeholder, such as sales, cost or profit. Instead, architects in our case used it to denote stable goal-like statements, such as the goal to achieve excellent service delivery, or to achieve a result-oriented working environment. Even after explaining the difference in one of our bi-weekly meetings, they kept using it the same way. An explanation of this could be that the concern concept is too general to be of use. What concerned the architects in our case was goals; so they used it to express goal-related concerns.

- The architects found it difficult to understand the difference between concern, goal and *assessment*. They sometimes used the assessment concept to store the contextual reasons for having a goal. For example, the goal of cost-reduction was annotated with an "assessment", that is a contextual reason, namely that the Dutch government faces the need for large budget cuts due to the financial crisis and the aging population.

4.4.3 Answers to research questions

Q1 *What constructs were used?* All constructs except the *conflict* relation were used by the architects in this case. The conflict relation was not used because the architects stated that there were no conflicting goals in this case. There is not much we can conclude from this: surely there are some cases where there are no conflicting goals, and we believe this is one of them; but there are other cases where there *are* conflicting goals. At the very least we can conclude that the idea of conflicting goals (goals that cannot always be all satisfied at the same time) was understood by the architects.

Q2 *Why (for which purpose) do they use these concepts and relations?* Q3 *Is this the intended use of the constructs?* The constructs of *stakeholder, goal, decomposition* and *contribution* were used as intended. The concept of *requirement* was not used as intended, but rather was used as if it were the same concept as that of a goal. That is, requirements were not always allocated to one architecture component.

The *means-end* relationship was used as intended, namely as relation from requirement to goal, because the tool did not allow any other use. The architects did not see a relevant difference with the contribution relation.

Finally, the concepts of *concern* and *assessment* were not understood by the architects.

Q4 *Which potentially useful constructs do architects miss in ARMOR?* The architects found it useful to express contextual reasons for a goal, and used the *assessment* construct to do this.

4.4.4 Validity

Our observations may have been influenced by the fact that we were involved with the definition of the language; this may have impacted the training positively (exceptionally inspiring explanations) or negatively (too much knowledge taken for granted). It may also have motivated the architects to have a socially desirable opinion about ARMOR. However, the architects had to do a real-world project with limited resources and as they are paying for this consultancy in money, and spending time on using ARMOR, they have no reason to present their experiences more favorably.

Also, we may have let his desire to have a usable and useful language influence his observations. This may have impacted the observations where architects where observed to use the ARMOR constructs as intended, but not the observations where the architects were observed to misunderstand the constructs of ARMOR. We regard at least those latter observations as credible. We also wish to emphasize that there was no 'golden standard' for determining if a concept was correctly used or not. The only standard we were able to use were the working definitions of the concepts [29]. We determined if a concept was correctly used or not. There was no budget available to hire a second observer. However, the results and selected examples were discussed with with the co-author of the original article and the authors of the language. Since, we were involved in defining the definitions, we were also the most suited in evaluating its use.

Finally, could we generalize from this case to other cases? Generalization from case studies cannot use statistical inference but can use reasoning by analogy [39, 111]. This means that we should explain our observations in terms of some general characteristics of the case, and provide a plausible argument that in cases with the same general characteristics, the same observations will be made.

Our observations all relate to understandability, and this relates to the cognitive competences of the enterprise architects in Organization 1. The architects in Organization 1 had to be able to design and understand a enterprise architecture for an organization of 3000 employees. Each of them had at least 2 years of experience as enterprise architect, and the organization operated its EA process at a maturity level comparable with level 2 of the US Department of Comments Architecture Capability Maturity Model. All of this may explain why they used the constructs of stakeholder, goal, decomposition and contribution as intended, and we expect that in other organizations, similar to Organization 1 in the aspects just mentioned, architects will understand and use these constructs as intended too. But we also expect that in many of those organizations, the constructs of hard and soft goal, requirement (as defined in ARMOR), concern and assessment will not be understood and be used in a way not intended by the designers of ARMOR, that the means-end relation will be considered superfluous and that negative contribution will not be distinguished from conflicts. This generalization is a hypothesis that must be validated in replications of this case study. We do not claim that it will be found to be true for all future case studies. However we do expect to encounter in the future cases similar to this one. This was a sufficiently strong reason for us to redesign the language.

4.5 Redesign

Figure 4.4 shows the meta-model of a stripped down version of ARMOR that we call Light ARMOR. We dropped the constructs of concern, assessment, hard and soft goal and means-end from the language as these were not understood, or the relevance not understood, by the architects. To facilitate recording contextual reasons for a goal (the construct missed by the architects in Organization 1), the *Goal* construct was extended with a text attribute in which this reason could be recorded in free text.

The construct of *Contribution* was replaced by that of *Influence* so that we can avoid the locution "negative contribution", which we ourselves find as confusing as the concept of negative income. A goal G1 *influences* another goal G2 if satisfaction of G1 has an effect on the satisfaction of G2. So influence is a causal relation.

We did keep the notion of *Conflict* as the inability to satisfy two goals simultaneously can be a case of causal prevention ("negative contribution") but it may also be a case of logical inconsistency, legal exclusion, ethical incompatibility, or plain monetary conflict (satisfying the goals jointly exceeds the budget). The concept of conflict is complex and awaits future exploration; but we find it too important to drop from the language just because it has not been used in one case.

Finally, requirements are a special case of goals, just as before, but we dropped the idea that we require a separate modeling concept for it. A requirement is just a goal assigned to a component of the architecture.

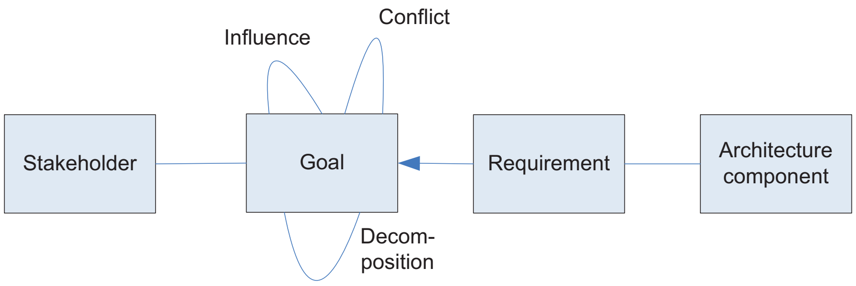


Figure 4.4: Meta-model of Light ARMOR

4.6 Case study 2

In addition to learning about the understandability of Light ARMOR, we would now like to learn about the utility of the language. Did our drastic reduction in the number of constructs impact the ability of enterprise architects to use the language (and supporting tool) to trace business goals to architecture components and vice versa?

The best way to find an answer to this question is to have enterprise architects use Light ARMOR to model the goals of an enterprise architecture, and then actually let them do the backward and forward tracing. This turned out not to be possible on short notice, and so we chose another form of action research, namely one in which the researchers themselves use their technique to solve a customer problem. In case study 2, we used Light ARMOR to solve an organizational problem following the engineering cycle of figure 4.2 and then used this experience to answer some validation questions about the design of Light ARMOR (figure 4.1). The research questions of case study 2 are, then:

- Q1 Is Light ARMOR understandable to architects?
- Q2 Can Light ARMOR be used to trace back and forth between business goals and enterprise architecture components?

4.6.1 Case description and research design

The case company, called Organization 2 henceforth, is at a drinking water production facility in the Netherlands. The company is responsible for the production and delivery of fresh drinking water to 1.2 million people and transports 73 billion liters of drinking water each year. It has about 500 employees divided over three divisions, viz. Production, Sales and Environment.

Enterprise-architects and information analysts in Organization 2 are facing rapid change and shrinking budgets and are looking for a technique that will enable them to assess the impact of changing business goals (forward tracing) and to determine the value of the architecture (backward tracing). We were given the opportunity to use Light ARMOR to link business goals to their current enterprise architecture model in a no-fee small consultancy project. This would allow them to see if they would want to use this technique in the future, and gave us the opportunity to perform a first test of Light ARMOR.

We planned and performed the following interactions with Organization 2. The first author interviewed the architect responsible for the EA of Organization 1, and studied primary documents documenting the EA and business goals. He designed a Light ARMOR model of the links with the two, and then interviewed the enterprise architect a second time, asking her, without providing training in Light ARMOR, (1) to explain the Light ARMOR model and (2) to assess whether she could use this model to solve her traceability problem. This provided the enterprise architect with sufficient information to conclude her problem solving cycle (figure 4.2) and provided the researcher with information to find initial answers to his validation questions (validation 2 in figure 4.1). The researcher kept a diary of his own modeling process and made a transcript of the interview to be able to answer his own research questions. We emphasize that in this case we interacted with only one enterprise architect of the organization.

4.6.2 Observations and explanations

- The major observation recorded in the researcher's diary is that it was often difficult to identify the stakeholders responsible for the goals from the primary documents or from the first interview with the enterprise architect. There are several possible explanations of this, such as that there is so much agreement about goals in Organization 2 that there is no need to record the goal owner; or that there is so much disagreement among the stakeholders that it is too dangerous to record a goal owner.

- The influence relation in this case is truly a causal relationship; including it in a model is an empirical statement that must be true about the world. For example, the goal to perform water filtering influences the goal to achieve clean drinking water. A second example is that the goal to achieve lower operating cost is influenced by the goal to achieve economics of scale with collaborative buying. Like all empirical statements, these influence statements could turn out to be falsified by events in the real world.

- The decomposition relation by contrast is not empirical, but definition. It was used to create a definition of a term that the stakeholders agreed on. It only expresses an agreement between those stakeholders and not necessarily between other stakeholders. For example, the goal to achieve excellent drinking water quality was decomposed into the goals of sufficient pressure, safe drinking water, odorless drinking water and visually clean drinking water. This is a definition that turns a soft goal into a hard goal.
- The architect judged that Light ARMOR could be used to link business goals to architecture components to realize forward traceability (assessing impact of goal change) and backward traceability (justifying an architecture component). She suggested that this would also be useful to link project goals to business goals, providing a way to scope projects.
- In the opinion of the architect, the conflict relation would be useful in the assessment of project risks. This would however also require a way to document the resolution of these risks. For example, record that one of the goals was dropped or that an other way was found to resolve the conflict.
- To test understandability of Light ARMOR we asked the architect to explain the model to us. The architect did not have prior training on GORE or Light ARMOR, but she could readily identify what the models meant.

4.6.3 Answers to research questions

The last observation provides support for the claim that Light ARMOR is understandable for practicing enterprise architects, which answers Q1 for this case.

The positive opinion of the architect about forward and backward traceability provides support for the claim of utility of Light ARMOR, answering Q2. In addition to the use for (1) estimating impact of change and (2) justifying the presence of an architecture component, the enterprise architect suggested using the model for (3) setting project goals and (4) documenting project risks and their mitigation. We will include these possible uses of Light ARMOR in our future research.

4.6.4 Validity

The major threat to internal validity is that the architect answered our questions in a socially desirable way. There is in this case nothing we can do to mitigate these risks, but in this case too we note that Organization 2 is looking for a way to exercise tighter control over its enterprise architecture in order to respond to changes in goals and a decreasing budget, and, doing so, has little reason to please the researchers. A *negative* response of the architect would have been really informative (and disastrous for the designers of Light ARMOR); the positive response that we actually received is less informative but is still encouraging.

The observations in this case make it plausible that if we were to repeat such a project in a similar organization (similar size, maturity of EA, experience of enterprise architect, dynamics of changing goals and shrinking budgets), we are likely to get similar results (positive opinion of the architect). This is a hypothesis to be tested in future case studies.

4.7 Lessons learned

We found that GORE concepts such as means-end relations and the distinction between hard and soft goals could not be used in our two case studies; and the concepts of concern and assessment taken from BMM and TOGAF could not be used either in our two cases. Also, the idea that a requirement exists, next to a goal, as a separate modeling concept puzzled the practitioners in case 1. They had difficulty distinguishing between the two.

Stripping these elements away and including the results from case study 2, we conclude that our case studies provide support to the claim that the GORE concepts of *stakeholder, goal, decomposition, influence* and *conflict* are usable in practice and potentially useful for the practitioner. The particular syntax of the language that we used in our case studies did not play a role in these evaluations.

A third lesson we draw from these two case studies is that a stripped down language adding only these elements to an EA language can be useful for maintaining traceability between business goals and enterprise architecture.

A fourth and final lesson is that the conflict relation can be confused with the negative contribution relation, but still can be useful to keep because it allows representing project risks and their mitigation. Risk in this case is limited to that the goals of the stakeholders are in conflict with each-other and if this conflict is not resolved, the project has no chance of succeeding. In

ARMOR the resolving of an obstacle is modeled through an assessment.

5

Second Evaluation[1]

[1]This chapter is based on an article in the proceedings of the International Conference on Advanced Information Systems Engineering[30]

5.1 Introduction

In chapter 3 we have defined an initial version of a goal-modeling extension to ArchiMate, called ARMOR. The context of our work is the ArchiMate language for enterprise architecture modeling [114]. ARMOR has been adopted in The Open Group standard for enterprise architecture modeling ArchiMate [114] as the motivation extension, through a standardization process we had no control over.

In chapter 4 we have provided an initial empirical validation of the usability and understandability of ARMOR, the language that was used as input to specify this official extension [36]. This validation showed that some users of the language experienced difficulty in understanding the extension to ArchiMate, and we proposed a simplification of the goal-oriented extension.

We present and analyze further data about understandability of goal-oriented concepts by enterprise architects, and we present explanations of the understandability issues. We present tentative generalizations about goal-oriented concepts in the context of enterprise architecture. We believe that the population of enterprise architects have no difficulty in using the stakeholder, goal and requirement concept. Regarding the relations, the influence relation is the best understood relation. These findings should also be true for the languages we based the motivation extension of ArchiMate on, namely i*, Tropos, KAOS and GBRAM.

We start with listing the research questions in section 5.2. Next we describe our research methodology in section 5.3. We detail our conceptual framework in section 5.4. Section 5.5 presents our data and analyzes the implications of these data for goal-oriented requirements concepts. In section 5.6 we provide answers to our research questions. In section 5.7 we discuss some implications for practice and for further research.

5.2 Research problem

We want to know how understandable the goal-oriented requirements extension to an enterprise architecture language is for practicing enterprise architects. So our population of interest is the population of enterprise architects, and in our research we investigate a small sample of them, and we investigate the understandability of the goal-oriented extension of the ArchiMate language. At the end of the chapter we discuss the generalizability of our results to the larger population of interest. Here we state our research questions:

Q1: How understandable is the motivation extension of ArchiMate by enterprise architects?

Q2: Which concepts are understood correctly? Why?

Q3: Which concepts are not understood? Why?

Q4: What kind of mistakes are made? Why?

We will define the concept of understandability, used in Q1, as the percentage of language users who understand the concept correctly. Q2 and Q3 ask which concepts are understood by all users or misunderstood by at least some users, respectively. For the concepts that are misunderstood, Q4 asks what kinds of mistakes are made. In all cases, we want to know not only an answer to the journalistic question what is the case, but also the research question why it is the case.

5.3 Research methodology

In terms of design research methodology, our empirical study is an evaluation of a technology implemented in practice, namely ArchiMate[125]. However, ArchiMate 2.0 had only recently been implemented at the time of the case studies and although it was used, it had not been used long enough on a large scale to make an evaluation by survey possible. Moreover, although surveys may reveal large-scale trends, they are inadequate at providing the detailed data about understandability that we need.

Our data comes from two groups of practitioners who followed a course on ArchiMate (in total two courses). Their homework provided the material we needed to assess understandability of goal-oriented concepts to enterprise architects, and to answer our research question above. The first group had 7 participants, and the second group had 12 participants. Their homework was an exercise based on an actual problem within their organization. These were real requirements engineering or EA design problems and therefore a fair representation of the difficulty level.

The participants of the two groups self-selected into the course, and so they may be more motivated or more talented than the “average” enterprise architect. They were also highly motivated to pass the course, since they were sent by their employer. They had to pass their homework exercises in order to get a certificate. Not passing the exam would have reflected badly on the subject and weakened their position in the organization.This would make understandability problems all the more telling.

All participants had at least 5 years of experience as an enterprise architect (or a similar role) and all had at least a bachelors degree (not necessarily in computer science or software engineering). The median experience is based on the linkedin profiles of the subjects. They have had some modeling experience, since this is common in their role of architect or business analyst. Since we did not do random sampling, and the groups are too small for statistical inference anyway, we cannot draw any statistical inferences from our results. We can only give descriptive statistics of our sample, but not draw statistical conclusions about the population of enterprise architects.

A controlled experiment would have given us more flexibility, but this is beyond our budget to do such an experiment with practitioners (i.e., we would have to pay them commercial fees).

However, because we have detailed data from the homework done by the participants, we will analyze possible causes for (mis)understanding goal-oriented concepts in ArchiMate, and then consider whether these explanations provide a reason for expecting (mis)understanding of goal-oriented concepts to occur outside our sample in a similar way that it happened in our sample. We will also compare our results with those in the published literature to see if results similar to ours have been found in other studies too, which would strengthen the plausibility of generalizations.

The assignments were relatively small compared to real-world enterprise architecture concepts. That reduces the generalizability of our results, but in a useful direction: we expect that in larger, real-world projects, understandability problems would increase, not decrease compared to what we have observed in our courses. This is useful because our findings provide suggestions for improvement of teaching and using goal-oriented concepts in enterprise architecture in practice.

Correction was done twice, and we assume that few mistakes in correcting the assignments have been made. A sample of the corrections of the student exercises have been discussed between the two authors (Engelsman and Wieringa) of the original paper where this chapter is based on, and no mistakes were found [30]. However, later we will see that even if we would increase or decrease the percentages (in)correct in the gradings with as much as 10 points, this would not change our explanations and qualitative generalizations.

5.4 Defining understandability

Many authors operationalize understandability in terms of the time needed to understand a model [50, 22] or the number of mistakes made in answers to questions about a model [61, 100, 91, 110]. Houy et al. [53] surveyed these definitions of model understandability and classified them in five types:

- Recalling model content. Subjects are given a model, and are given time to study the model. Afterwards they have to recall how the model looked like.
- Correctly answering question about model content. Subjects are given a model and are given time to study the model. Afterwards they are presented with a questionnaire and have to answer questions about the information in the model (e.g. the constructs used in a model).
- Problem solving based on the model content. Subjects are given a model to analyze, and are asked to solve problems (answer questions) based on this model. For example, if the model were a route for a bus, they were asked questions about the route of the bus.
- Verification of model content. Subjects are given a model and a textual description. They have to answer questions regarding the correctness of the model content based on the problem description.
- Time needed to understand the model. Subjects are given a model to study. The time needed to answer questions about the model is measured.

Another interesting approach to measuring understandability, not mentioned by Houy et al., is that of Caire et al. [19], who measured the ability of subjects to guess the definition of an i* construct by looking at the icons.

All of these measures indicate a passive form of understanding because they concern the understanding of a given model. We are however interested in a more active kind of understanding of a modeling language, this is needed when an analyst uses the language to build models. Such an active concept of understanding is used by, for example, Carvallo & Franch [20] and by Matulevičius & Heymans [79], who measured the number of mistakes made in constructing i* models, and by Abrahao et al., who measured the time needed to build a model [2].

We define the understandability of a concept for a set of language users in this chapter as the percentage of language users who, whenever they use the concept when building a model, use it correctly. Understandability is thus relative to a set of language users. In this chapter we measure the understandability of goal-oriented concepts in ArchiMate 2.0 in a sample of 19 language users. From our observations, we draw conclusions about understandability of the GORE concepts of ArchiMate, and of goal-oriented concepts in general, for enterprise architects.

5.5 Data analysis

Table 5.1: Understandability of goal-oriented concepts in ArchiMate by a sample of 19 practitioners. Row *i* column *j* shows the percentage of times that practitioner *i* used concept *j* correctly.

Practitioner	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	avg
Stakeholder	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
Driver	66,6	100	100	100	77	na	69	50	38	100	55	100	100	100	69	33	100	85	100	47
Assessment	25	8,3	100	44	100	na	13	50	100	100	100	71	100	100	83	90	100	97	100	47
Goal	94	82	100	95	100	92	98	100	100	50	100	100	100	100	100	100	100	100	100	68
Requirement	100	100	100	75	100	na	100	100	0	80	100	91	62	100	100	100	100	95	85	57
Decomposition	0	na	na	100	100	83	24	na	62	na	100	100	100	50	na	na	79	57	15	26
Influence	100	50	na	100	100	100	100	na	100	na	100	100	100	100	100	100	100	100	100	79
Realization	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100

Table 5.1 summarizes the scores that the 19 enterprise architects received on their homework. The numbers are the percentage of correctly used concepts by each subject. When a subject did not use a concept at all, the corresponding cell contains “na”. Subject 1-7 are the subjects from 2011 and subject 8 - 19 are the subjects from 2012. The avg column shows the percentage of users that always uses the concept correctly. Looking at this column we see that only four concepts were used correctly by at least half of the subjects: the concepts of stakeholder, goal, requirement and influence. We now discuss our findings in detail.

5.5.1 Description of model complexity

In total the 19 subjects constructed 246 diagrams and on average the models contained 9 concepts. However, complexity of the models varied. Some diagrams contained as little as 2 concepts, and others contained 35 concepts. Not every diagram contained every concept. This is because ArchiMate uses views to reduce model complexity. There are roughly three kind of views. The first is a stakeholder view, showing the stakeholders, drivers, assessment and initial goals. The second is a goal refinement view showing the modeling of goals, goal influence, goal decomposition and goal realization through requirements. The third view shows the realization of requirements by architecture components. Figure 5.1 shows the frequency with which different concepts were used. The most frequently used concepts are those of goal, stakeholder and assessment.

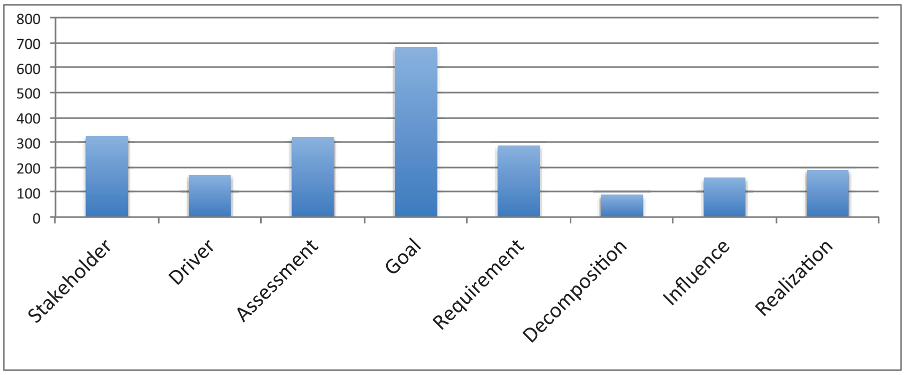


Figure 5.1: Frequency of use of goal-oriented concepts in 246 EA models

5.5.2 Analysis of GORE concepts and relations in ArchiMate

Stakeholder The first concept under analysis is the stakeholder concept. This concept is based on definitions from TOGAF, i* and Tropos. TOGAF defines a *stakeholder* as an individual, team, or organization (or classes thereof) with interests in, or concerns relative to, the outcome of the architecture [115]. This seems more general than the definition of actor in i* and Tropos as entities with intentional properties such as goals, beliefs, abilities, and commitments [129]. The motivation extension of ArchiMate adopted the more general definition of TOGAF [29, 73].

In our experiments the stakeholder concept was perfectly understood by every student. There was not a single mistake made in all instances of use. This can be explained by the fact that the TOGAF stakeholder concept is a well known concept by the subjects. Its definition is very clear, and it is substantially different than the other concepts used in the motivation extension, so that it is not easy to confuse the stakeholder concept with any other concept. For these reasons we think this finding will generalize to other ArchiMate users too. To the extent that the concept of actor in i* and Tropos coincides with that of TOGAF stakeholder, we expect that users of i* and Tropos will find the actor concept unproblematic too, and to be able to use it without mistakes.

Driver The driver concept is not found in the GORE literature, but used in the EA literature. TOGAF defines driver as the key interests that are crucially important to the stakeholders in the system, and determines the acceptability of the system [115]. The motivation extension of ArchiMate adopted this definition[29, 73]. In our experiments only nine subjects (47%) understood the driver concept correctly. The most common mistake made with driver was that it was used as a goal. For example, subjects modeled the goal 'to improve Financial Information' as a driver. But the driver corresponding to this goal is the key interest 'Financial Information'. Apparently the definition of driver is not very clear, and it is so close to the concept of a goal that it generates more confusion than clarity. We see no reason why this would not confuse other practicing enterprise architects, so we think that this finding will generalize to other ArchiMate 2.0 users as well.

Assessment The concept of an assessment too is based on definitions found in the EA literature. The Business Motivation Model (BMM) defines an assessment as a judgment about some influencer that affects the organization's ability to employ its means or achieve its end [16]. The motivation extension of ArchiMate attempted to make this more concrete by defining an assessment as the *outcome of the analysis of some driver* [29, 73]. In our experiments nine subjects used the concept perfectly. The most common mistake was that the assessment concept was used as a goal. For example, correct use of an assessment would be 'the financial information is incorrect'. This is a possible outcome of an analysis of the key stakeholder interest 'financial information'. However the subjects used the concept often to denote a goal like 'improve financial information', just as we saw with the driver concept above. The distinctions between a key interest, an analysis of a key interest, the outcome of the analysis, and the goal motivated by this outcome, were lost on most of our subjects. Moreover, the outcome of an analysis of a key interest can indeed be to 'improve something'. For these reasons we think this confusion will be present in other enterprise architects who use ArchiMate, as well as in users of the assessment concept in BMM.

Goal The ArchiMate definition of goal is based on a combination of the EA literature and the GORE literature. KAOS defines goals as desired system properties that have been expressed by some stakeholder(s)[119]. This seems more technical and solution oriented than the i* and Tropos concepts of a goal as the intentions of a stakeholder i*[129]. BMM defines goal as a state or condition of the enterprise to be brought about or sustained through appropriate means BMM[16]. ArchiMate defines goal as *some desired end that a stakeholder wants to achieve*[29, 73]. In our experiments 13 subjects understood the goal concept perfectly. The most common mistakes made were that a goal was either used as a driver or as a requirement. For example, the subjects would write down 'financial information' as a goal, but it should actually be something like 'improve financial information'. When it was used as a requirement, it was written down as 'the system should have 100% availability'. We can reuse our explanation that the distinction between driver, goal and requirement were lost by the subject in that instance. The concept of goal can therefore be understood and used by practicing enterprise architects, but mistakes are made too. Since the ArchiMate concept of a goal is similar to that in KAOS, i* Tropos and the BMM we expect that this will happen in users of those languages too. This calls for clearer guidelines in the application of the goal concept.

Requirement We based our definition of the requirements concept on the GORE literature. In KAOS a requirement is a goal assigned to an agent of the software being studied [119]. In GBRAM a requirement specifies how a goal should be accomplished by a proposed system [7]. The ArchiMate motivation extension defines requirement as *some end that must be realized by a single component of the architecture* [114]. In our experiments in total 11 subjects (57%) perfectly understood the concept. In general the requirement concept was reasonably well understood. This can be explained by the fact that it is a well known concept already known in practice by the subjects.

However, there were quite a large number of mistakes. Many subjects specified requirements that were goals not yet allocated to a system. For example, instead of the 'the system should have a financial reports function', they specified the goal 'improve financial reports'. We see again that semantically close concepts are confused by practitioners, even though the definitions of the concepts are clear. We expect this confusion to be present in other users of ArchiMate as well.

The decomposition relation ArchiMate 2.0 based this relation on a combination of concepts from the EA and GORE literature. i* defines a decomposition as an element that is linked to its component nodes. [129]. BMM uses a similar definition, but it is more aimed at goals. BMM defines decomposition as an end that includes an other end BMM [16]. In Tropos, a parent goal is satisfied if all of its children goals are satisfied[13]. In KAOS the conjunction of all the sub-goals must be a sufficient condition entailing the goal KAOS [119].

The motivation extension of ArchiMate defines decomposition as a *some intention that is divided into multiple intentions.*[73]. This was understood correctly by only five subjects (26%). When the decomposition relation was used incorrectly, it was used as a influence relation, which in ArchiMate is defined as a contribution relation. For example, correct decomposition of the goal 'improve correctness financial information' should be 'improve correctness financial information regarding outstanding debt and improve correctness financial information sales'. This decomposes a goal into more detailed goals. However, many subjects decomposed the goal 'improve correctness financial information' into the component goal 'acquire a financial reports system that records sales information'. But this is an influencer, i.e. a new goal that contributes to the original goal. The confusion is probably caused by the fact that satisfaction of an influencer increases the satisfaction of the influenced goal, just as satisfaction of the components increases the satisfaction of the composite goal. Based on this analysis we expect other users of ArchiMate to have similar problems.

The influence relation In i* a contribution is a link of elements to a soft goal to analyze its contribution[129]. Tropos defines contribution analysis as goals that can contribute positively or negatively in the fulfillment of the goal to be analyzed [13]. ArchiMate defines this as a goal G1 contributes to another goal G2 if satisfaction of G1 influences the satisfaction of G2 positively or negatively [29, 73]. The influence relation was understood correctly by 15 subjects (79%). In the cases were the relation was not used correctly, the subjects linked requirements with goals on a 1 on 1 basis, which amounted to stated the same goal twice. Others used a standard ArchiMate association relation where they should have used an influence relation. To further reduce the misunderstandings of the influence relation, better guidelines must be found.

The realization relation This relation is based on relations found in the GORE literature. i* defines a means-end relation, which is a relation between an end and a means [129]. KAOS defines a relation for linking requirements to operations [119]. The ArchiMate motivation extension defines the realization relation as a relation that some end that is realized by some means [29, 73]. All subjects used the realization relation correctly. This can be easily explained by the fact that the support tool only allows connecting a requirement to a goal and an architecture element to a requirement so that the relation cannot be used incorrectly. 100% correct use therefore has no implication for (mis)understanding of the concept by tool users.

5.5.3 Sample model created by a subject

We have provided an example model with the types of mistakes a subject made in figure 5.2. In this example we have stakeholder X. The stakeholder was modeled correctly. The subject identified two drivers. The first to prevent incorrect usage of the videos and to redesign the operational activities in a lean model. These two drivers are stated as a desired state, and therefore should be a goal. Second, we have two assessments prevent reputation damage and to facilitate employees in their work. These two should have been modeled as a goal, as they are desired states. Finally, the subject identified a goal independent employees. This is stated how a driver should have been stated.

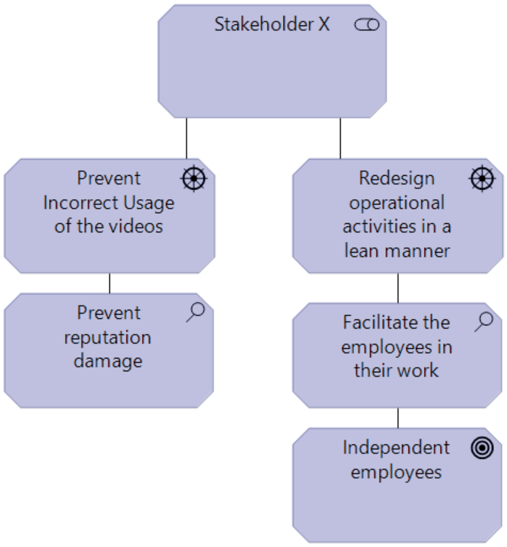

Stakeholder X
Prevent Incorrect Usage of the videos
Redesign operational activities in a lean manner
Prevent reputation damage
Facilitate the employees in their work
Independent employees

Figure 5.2: example goal model created by a subject during the assignments. This model has been redrawn in the latest version of ArchiMate.

5.6 Answers to research questions

Q1: How understandable is the motivation extension of ArchiMate by enterprise architects? As shown by the last column of table 5.1, not all of the motivation extension is understood very clearly.

Q2: Which concepts are understood correctly? Why? Only the stakeholder, goal, requirement concepts and the influence relation were understood by the majority (scoring more than 55%). However the requirement concept was a borderline case where a lot of mistakes were made. Our explanation of this level of understanding is that they are well known concepts already used in practice, and that they have a semantic distance that prevents confusion. However, the distance between requirement and goal is smaller than the other concepts and immediately we saw an a drop in understandability.

Q3: Which concepts are not understood? Why? The concepts of driver, assessment and decomposition were not very well understood. They were often confused with other concepts, such as that of a goal. Our explanation is that drivers, assessments and goals are very closely related and may even overlap, and that the definition of the decomposition relation overlaps with the definition of the influence relation.

Q4: What kind of mistakes are made? Why? The subjects made two types of mistakes. Drivers and assessments were modeled as goals. A driver is related to a stakeholder and an interest area of the stakeholder. A goal is a statement of desire about this interest area. This makes goals and drivers conceptually very close and created confusion in our subjects.

The same is true for the assessment concept. An assessment is the outcome of some analysis. It is not defined what this outcome should be. It can very well be a goal or something else, which is confusing again. The use of the requirement concept to model a goal is similar. Because both concepts are closely related, the difference between desired functionality from the viewpoint of a stakeholder is very much similar to the stated functional need of a system. The only difference is the perspective.

The second type of mistake is that an influence relation was expressed by means of a decomposition relation. Again, the definitions turn out to be too close to each other for many of our subjects.

5.7 Discussion

5.7.1 Generalizability

To which extent are our results generalizable beyond our sample of practitioners? In our experiments every subject had at least five years experience, the minimal of a bachelors degree. Enterprise architects usually have the same educational background as our subjects. Our subjects were responsible for translating business strategy and business goals into requirements models and they had to design an enterprise architecture based on these requirements. This is similar to the tasks enterprise architects have to perform in general.

Moreover, the results from this study match with our previous research. In our previous work [36] we reported about a real-world project in which practicing enterprise architects used ArchiMate to redefine an enterprise architecture and link it to changed business objectives. They used the stakeholder and goal concepts as intended. They had some trouble understanding the requirement concept and often formulated requirements as if they were business goals. We also saw that the subjects had a difficult time to see the difference between goals and drivers. The driver concept was too general to use for the subjects. The same was true for the distinction between driver, goal and assessment. Those finding and their explanations agree with the ones reported about this in chapter 4.

All of this justifies the claim that other enterprise architects may understand and misunderstand goal-oriented ArchiMate concepts in the same way as our subjects did. This is a weak generalization, as it says “this can happen more often” without giving any quantification how often it could happen [43]. But such a quantification is not needed to draw some implications for practice, as we do below.

Because the goal-oriented concepts that we used have been taken from other existing goal-oriented languages, we hypothesize that our conclusions may be generalized to those languages too. Again, we cannot quantify this beyond the weak claim that this may happen in those languages too. But we do claim that our findings are sufficiently generalizable to motivate similar research for those languages.

5.7.2 Validity

Construct validity is the extent to which theoretical constructs are applied and measured correctly in our study. The only theoretical construct that we use is that of understandability, and we defined it in section 5.4. Our definition agrees with that used by other authors [20, 79] but with that of all other authors. Our definition refers to the number of mistakes made when building models, and not the the amount of time (indicator of effort) required to build the models. Other definitions refer to the number of mistakes or the amount of time needed to answer questions about the models. Comparison of our results with that of studies that use another definition of understandability should be done with caution.

Internal validity is the support for our causal explanations of the phenomena. Could subjects have misunderstood some concepts for other reasons than the ones we hypothesize? For example because they lack competence or because they were explained badly in the training? We cannot exclude these other explanations, but find them less plausible because all subjects had similar background and experience, and because the teachers similarly have several years of experience teaching these concepts. and even if these explanations were true for some subjects, this would not invalidate our explanation in terms of semantic closeness of concepts.

External validity is the support for generalization from our quasi-experiment. Because our explanations do not refer to particular properties of our sample but are stated in terms of the language itself, and because other practitioners are relevantly similar in background and experience to those in our sample, we think our conclusions are generalizable. But we do not claim that they are generalizable to the entire population of practicing enterprise architects, nor to all other goal-oriented languages.

5.7.3 Implications for practice

ArchiMate 2.0 is now an Open Group standard, and the concepts we investigated in this chapter will remain present in the language. However, one practical implication of this chapter is that in future training programs we will not teach all concepts anymore. We will make a distinction between the recommended minimal concepts and less important concepts. We will recommend that future users of the language at least should use the stakeholder concept, the goal concept and the requirement concepts.

A second implication is that we need more practically usable guidelines for the use of the concepts that we do recommend, because other than the goal and realization concepts, we expect that many practitioners will misunderstand and incorrectly apply the basic concepts of goal and requirement and the relations of influence and decomposition. This is a topic for future research.

6

Literature Review: Understandability of GORE

6.1 Introduction

Goal-oriented requirements engineering (GORE) modeling languages have been around for over twenty years [23, 130]. However, transfer to practice so far has been very limited [79]. The leading notations are KAOS and i*. Previous research showed that these GORE languages are very rich in notational concepts and therefore possibly difficult to understand.

In our research (chapters 4 and 5) we experienced a similar issue [36, 30]. Earlier, we defined the ARMOR goal modeling language (chapter 3), which contains the major constructs from other goal-oriented requirements engineering languages [29]. ARMOR has been included in the Open Group enterprise architecture modeling language ArchiMate and is currently used in the practice of enterprise architecture modeling [115].

In order to improve ARMOR, and to improve our teaching methods of goal-oriented concepts to practitioners, we need to better understand the cause(s) of these understandability problems. Hence, we want to know which explanations and explanatory theories can be found in empirical evaluation studies of GORE languages. In this chapter we summarize what is known about this from the literature.

During this research we found out that there is little known about GORE understandability. There are only a few empirical evaluations of GORE and none of them provide the theoretical explanations we seek. Therefore we extended our research to include evaluations of UML regarding understandability, in the hope that results from that neighboring field will be relevant for the empirical study of GORE languages as well. It is not our goal to identify all the evaluations of all languages, but to find sufficient explanations to improve our experiments and teachings.

We first state our research problem more precisely in section 6.2. Next, in section 6.3 we describe our research methodology. Section 6.4 provides an overview and an analysis of our results. Section 6.5 summarizes our answers, discusses validity, and lists some lessons learned from this research. We provide a detailed listing of the results in section 6.6.

6.2 Research Problem

We start with a conceptual framework for evaluating understandability. Understandability has been long investigated in the science of user interface design, where it is part of usability. The different indicators for understandability used in the literature turn out to be indicators of usability, and we therefore

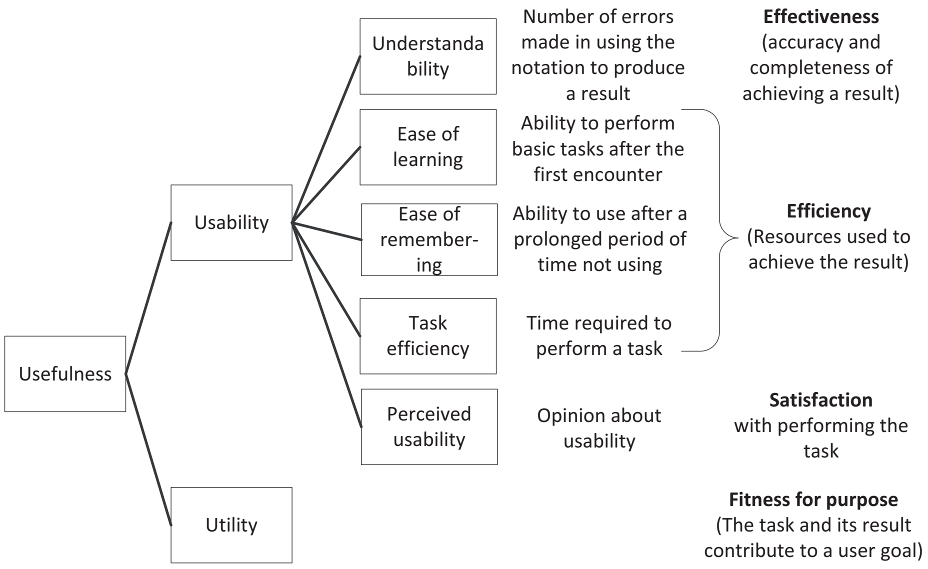

Usefulness
Usability
Utility
Understanda bility
Ease of learning
Ease of remember-ing
Task efficiency
Perceived usability
Number of errors made in using the notation to produce a result
Ability to perform basic tasks after the first encounter
Ability to use after a prolonged period of time not using
Time required to perform a task
Opinion about usability
Effectiveness (accuracy and completeness of achieving a result)
Efficiency (Resources used to achieve the result)
Satisfaction with performing the task
Fitness for purpose (The task and its result contribute to a user goal)

Figure 6.1: The conceptual framework.

use a conceptual framework that we borrow from usability studies. Lauesen [74] summarizes the framework for usefulness shown in figure 6.1.

Effectiveness is the accuracy and completeness of achieving a result with the notation; this is explained further below. Efficiency is the amount of resources used to achieve a result, which is usually decomposed into ease of learning, ease of remembering, and time to produce the result. Satisfaction is the opinion of the user about usability.

In this framework, understandability of a notation is effectiveness of the notation, which is the accuracy and completeness of achieving a result with the notation. Houy et al [53] identified six different ways to operationalize this.

- *recalling model content.* This is tested by giving a subject a certain amount of time to study a model and then asking the subject questions about which elements are in the model, measured with a correctness percentage.

- questions about model content and the percentage correct answers is taken to indicate understandability.

- subjects are asked to solve problems using the model (e.g. to determine the route of a bus), and the percentage correct answers is measured.
- subjects are asked to verify whether certain parts of the model are a correct representation of the problem domain. Measurement is again by correctness percentage.
- the time needed to complete a certain task is taken as indicator. In the framework of figure 6.1, this is actually task efficiency and not understandability.
- perceived understandability. In our framework, this corresponds to satisfaction.

In our previous work [30] we have used yet another understandability concept, which is the number of errors made when *using* the notation to build models. This contrasts with the operationalizations above, in which the task is to *read* a model.

Understandability in the literature can thus mean different things, and we have to take this into account if we compare research results.

6.2.1 Research questions

Based on this framework and our research goal we can identify the following research questions:

Q1 How is understandability operationalized in usability experiments?

Q2 Which studies have been performed to measure understandability of GORE and UML notations? (The first version of the research question only asked about GORE notations.)

Q3 Which explanations can be found in the literature regarding understandability phenomena of GORE and UML notations?

Q4 Which theories are used in these explanations?

6.3 Research Methodology

The data was gathered by performing a literature study. We have based our work on the procedures defined by Kitchenham [66].

Table 6.1: Summary of search strings

Search string
Empirical evaluation understandability goal-oriented requirements engineering
Empirical evaluation understandability goal models
Empirical evaluation understandability KAOS
Empirical evaluation understandability TROPOS
Empirical evaluation understandability i*
Empirical evaluation understandability of UML

The search for literature was done with Web of Science, Scopus and Google Scholar. Table 6.1 lists the used search terms. We have used each search string individually in the search engines and used the default AND operator. We made an initial selection of the required papers based on the title, keywords and abstract. We selected our sample by defining inclusion criteria.

We initially started with only the GORE languages. However, these provided us with insufficient results, therefore we extended our search to UML as well. So we are interested in papers that perform experiments regarding understandability of GORE and UML notations.

During the analysis of the results we looked at the goal of the study, the operationalization of understandability, the sampling method of the experiment, the observations, and possible explanations of the results. Results not complying were immediately dismissed. A first selection resulted in 47 papers. Further applying the criteria by reading the papers reduced the list to 21 papers. Not all papers found where directly evaluations of understandability. Often they were also of utility or understandability of artifacts based on UML models.

Tracing the papers into subsequent studies revealed an additional paper.

6.4 Results

Section 6.6 lists the detailed comparisons and figure 6.2 summarizes the main observations made in GORE understandability experiments. Here, we list our own main observations about these experiments.

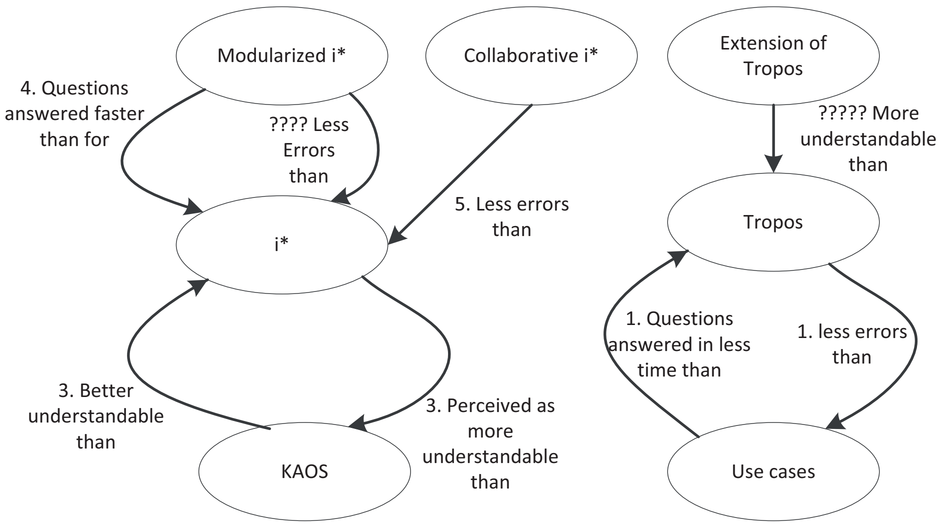

Modularized i*
Collaborative i*
Extension of Tropos
4. Questions answered faster than for
???? Less Errors than
5. Less errors than
????? More understandable than
i*
Tropos
1. Questions answered in less time than
1. less errors than
3. Better understandable than
3. Perceived as more understandable than
KAOS
Use cases

Figure 6.2: Summary of GORE results.

Scope of the study. We have observed that there is at the time of writing a lack of empirical validation regarding understandability for GORE notations. We were only able to find seven papers (including our work). One [50] of these provided theoretical explanations for the observations made in the experiments. However, this theory is aimed at why diagrams are better than textual alternatives and no use in explaining understandability issues of diagrams. We therefore extended our search to UML understandability experiments, hoping to find theories relevant for our own research.

Operationalizations. We found that the most frequently used operationalization of understandability is the percentage of answers answered correctly.

The second most frequently used operationalization was that of task completion time, which in the usability framework of figure 6.1 indicates task efficiency, which is a part of usability of the notation.

The third most frequently operationalization is perceived understandability. Subjects were asked if they thought a language was understandable, however according to [74] there is little correlation between subjective satisfaction and objective performance.

Sampling. Most of the subjects were students. Two of the studies (numbers 7 and 20) used practitioners. All samples were not randomly selected. Non-random selection limits the possibilities to do statistical inference from the data. If treatments were compared, then allocation of the treatments to subjects was randomized.

Explanations. Not all papers explain their observations. Of the 22 reviewed papers, we found 12. that provided explanations. Three kinds of explanations are given in the literature:

- Experience
- Design flaws with languages.
- Badly designed supporting materials

The first explanation is that of experience: The more experience a user has, the higher the ability to understand the language. This was found by Hadar et al, [50], Soh et al [107], and Cruz et al [22]. This explanation goes both ways: Someone who does not understand a notation will avoid using the notation, and if he or she uses it, will stop using it very soon. Conversely, repeated use of a notation that one understand reasonably well, will produce increased understanding.

The second explanation authors give is is that there are design flaws with the languages, e.g. are too many concepts, semantic definitions are unclear, or the visualization of the constructs is ill designed. This was found by Matulevicus and Heymans [79], Purchase et al [100] and Siau and Loo [105]. This was also the explanation we used in [36] and [30].

The third explanation given in the literature provides is is that supporting materials are badly designed. Several authors mention that languages are plagued with bad tooling and bad training material [20], [79] and [105].

We add to these an additional explanation of some of the findings.

We often saw that a variant of a language was defined to fit in a certain problem domain, such as requirements for collaborative systems [113] or self-adaptive systems [90]. The experiment then always shows that the extended language outperforms the original language on the domain for which the extension was defined. For example, Teruel et al. adapted i* to work in CSCW systems. The explanation of this finding is then that the extension contains concepts that makes it easier to express requirements for this particular kind of system.

Several authors did not provide explanations for their observations. This is troublesome, because without properly identified explanations it becomes hard to generalize the results of the study to other experiments.

Mate et al [78] provided no explanation why a modularization extension of i* was easier to understand than vanilla i*. One explanation that we propose, is that a modularization extension provides an abstraction mechanism that reduces the load on the short term memory of its users. This explanation would be a basis for generalizing to similar cases. For example, The explanation can be used to the study of [71] Lange and Chaudron. They investigated if an interactive view for UML improved understandability of UML models. They observed that by providing an abstraction layer the subjects could find information faster and they answered more questions correctly. Modularization provided an abstraction mechanism that reduced the load on the short term memory of its users.

Kamsties et al [61] provide no explanation why SCR specifications were easier to understand than UML specifications.

Kuzniarz et al [69] provide no explanation why UML diagrams with stereotypes are better to understand than UML models without stereotyping. This can be explained by the hypothesis that Nugroho [91] (study 18) explained his results, namely that less details allows interpretation errors. The same explanation is also applicable to the findings of Reggio et al. [103]. They found that a heavy weight version of activity diagrams was easier to understand than a lightweight version. They do not provide an explanation for this, but we can now see that the heavier version contained more information than the lightweight version, which may have made it easier for the subject to answer questions about the models. More information was explicitly specified in the diagrams.

Theories. We only found two papers that gave a theoretical basis for its explanations. Hadar et al [50] (study 1) listed a single theory why Tropos models were more comprehensible than use-case models. They referred to cognitive science theories, where it states that it is easier to retrieve information from a diagram than from text. However this theory is of no use for us, since we are only interested in understandability of graphical notations, not for textual notations.

Siau et al [105] (study 17) Listed three theories to explain why people find the UML difficult to understand.

The first theory is that new knowledge is always assimilated to existing knowledge in the human brain. This can be found in the information processing theories. If subjects were exposed to anything other than object-orientation before learning UML, this would hinder understandability, for example exposure to a functional paradigm would hinder understandability of UML (which is object oriented).

This relates to one of the categories of explanation that we identified above, namely that subjects with more experience understand a language better. Increased experience creates the constructs in the subject's mind that help to assimilate the constructs of the language.

The second theory given by Siau et al. is that languages with a large number of concepts are difficult to learn. This is to the fact that the humans are constrained by the size of their short term memory [84]. This further substantiates the second category of explanation identified above, namely that language design flaws create problems with understanding.

The third theory given by Siau et al. is based on human cognition and human learning literature: people find it difficult to learn something without assistance. Without proper assistance during the learning process the learning curve for the UML is too steep for most people. This gives further support to our third explanation identified earlier: To learn a new notation, proper support is needed.

6.5 Answers to research questions

The answer to Q1 is that understandability is operationalized as a correctness percentage, the time needed to answer questions and as a subjective evaluation of understandability.

The answer to Q4 is that we found three explaining theories. Humans find it difficult to learn new knowledge unassisted, humans map new knowledge to existing knowledge and humans find it difficult to work with a large number of concepts.

Based on the explanations provided we can identify a number of implications.

- We need to increase the experience of the users during our courses. This would mean either to allocate more time to modeling assignments or more sessions in our courses. In future studies, in a pre-test, subject experience should be measured, in order to better be able to interpret experimental outcomes.

- We need to look at the semantic definitions of our language and make sure there is sufficient distance between the concepts. This implication is also valid for other GORE languages like i* and KAOS and even to the UML.

- We need to improve our study material. This means that designing training material should be itself a design problem as important as designing the language. If training material is of importance to the understandability results, then a more detailed description on how the subjects were trained (or a more standardized) way is important to be able to compare the findings of different understandability studies.
- We need to accurately capture the existing knowledge of course participants, in order to personalize the course offering. However, for other studies this does mean that capturing existing knowledge is important, this again to improve comparing understandability findings to each other.
- This means that in order to get an improved understandability we need to focus more on the core concepts during the courses. Other conceptual languages should follow a similar path to improve understandability.
- In the future we need to improve the learning process by assisting participants more closely. For future understandability studies, more attention needs to be given on how intensive subjects were trained. This to be able to better compare the different studies.

6.5.1 Future research

In future validation studies we will incorporate the results from this analysis. The first step is to evaluate if more experience leads to better goal models. We will organize a workshop at REFSQ 2014. We believe that the population of REFSQ 2014 contains more experienced users of goal modeling languages. We will record subject experience as well.

It is impossible for us to change the definitions of the goal modeling constructs in ArchiMate 2.0; however we can propose a subset of core concepts which have sufficient conceptual distance to improve understandability.

Furthermore, as soon as we resume our validation studies with practitioners, we will focus on improved training material, and we will assist the participants more closely.

6.6 Detailed Listing of Results

Table 6.2 provides an overview of the identified and used papers.

	Research Goal	Operationalization	Sampling	Observations	Author explanations
1	To compare the comprehensibility of Tropos and use-cases [50]	For Tropos and UC the comprehension level (counting the correct answers to questions about model content), effort (time to complete all questions) and productivity (comprehension level / effort * 100 were measured.	Samples of students. 19, 31 and 29 subjects. All non random sampling. But random assignment to groups.	Tropos models were more comprehensible than use-case models. It took more effort to answer questions about Tropos. Productivity is equal for both languages.	They explain the results by cognitive science theory. Diagrams are better suited for searching and inference
2	To present an experience report on comprehensibility of i* [20].	Comprehensibility is measured by errors in the constructed models.	Non-random sampling. No details on sample size.	i* models were difficult to build and modify	i* diagramming tools are not suited to perform the experimental tasks.
3	To determine if i* or KAOS is a better language [79].	Error frequencies in constructing or reading models and the ability of the users to understand the concepts of the language; Opinion of the subjects.	19 students, non random sampling.	i* and KAOS had equal icon understandability but KAOS had better understandable concepts in use. The subjects found i* easier in use in practice.	KAOS uses identical shapes for different concepts; KAOS is too rich in concepts.

	Research Goal	Operationalization	Sampling	Observations	Author explanations
4	To determine if a modularization extension of i* improves understandability over normal i* [78].	Time spent in seconds to answer questions, correct answers about model comprehensibility	28 subjects, unknown type or sampling.	The new version of i* reduced the time spent on answering questions and reduced the error rate percentage	No explanation,
5	To determine if an extension of i* is better understandable than normal i* when used for modeling collaborative systems [113].	Correct answers in an questionnaire.	Samples of 30, 45,9 students. Non random sampling	The extension improved understandability for this task	Special constructs for collaborative systems were added.

	Research Goal	Operationalization	Sampling	Observations	Author explanations
6	To determine if an extension of Tropos for self-adaptive systems is more understandable than Tropos itself in the context of modeling self-adaptive systems [90].	Correctness of the answers to comprehension questions.	Samples of 12,6 students; non random sampling	This variant of Tropos was more understandable than normal Tropos for this kind of task	Tropos was extended with constructs that were needed for this task.
7	To determine if a GORE extension for enterprise architecture is understandable [36, 30].	Correctness of modeling diagrams	Sample of 19 practitioners; non-random sampling	The subjects only understood a limited set of concepts	The subjects could not distinguish between all the concepts because of semantic unclarity and semantic distance of the concepts.

	Research Goal	Operationalization	Sampling	Observations	Author explanations
8	To determine if a heavy-weight (more precise) version of activity diagrams is better understandable than a light-weight version [103].	Correct answers to comprehension questions; the time needed per answer in minutes.	Samples of 62,26 students. Non random sampling.	The heavy-weight version was better understood than the light-weight version.	No explanation.

	Research Goal	Operationalization	Sampling	Observations	Author explanations
9	To determine if DFD's are better understandable than UML use-case models [10].	Correct answers to comprehension questions; time (in minutes) to complete a question; perceive comprehensibility (opinion); errors in constructed models; time needed to create a model; opinion about modeling method.	53 students, non random sampling	UC models were faster to understand. UC models were of a higher quality. Subjects found DFD models easier.	i) The errors were in data flow modeling. Since UC does not contain data-flows the errors in this section were lower. ii) UC models contain less concepts and therefore are less confusing. iii) Because DFD diagrams looked complicated, they must be more comprehensible.(Lauesens second law)
10	To compare the understandability of UML sequence diagrams and UML collaboration diagrams [68].	Correct answers to questions about the models.)	124 students, non random sampling	There was no difference between the techniques.	The scenarios about which the students had to answer questions were possibly incorrect. Any difference in understandability is compensated for by increased effort by the subjects

	Research Goal	Operationalization	Sampling	Observations	Author explanations
11	To compare SCR [51] with UML specifications [61].	Time to complete all the questions about the model; correctness of the answers about the models.	22 students, non random sampling. Allocation of tasks to subjects was random .	Less errors in SCR questions; response times for SCR were faster.	SCR is easier to understand. No further explanation.
12	To investigate whether user preference for UML sequence or collaboration diagrams is reflected in improved accuracy in understanding [14]	Correctness of the answers about model content measured through a questionnaire.	124 students, non random sampling.	There was no difference	No explanation.

	Research Goal	Operationalization	Sampling	Observations	Author explanations
13	To determine if UML models with stereotyping are better understandable than models without stereotyping [69].	The number of correct answers for each subject and the time which was required to answer all the questions. The measurement instrument was a questionnaire.	44 students. Non random sampling.	The introduction of stereotyping improves understandability.	No explanation.
14	To determine if UML state chart diagrams with composite states are better understandable than those without them [22].	The number of correct answers in a questionnaire.	Samples of 55,178,14,13,24 (students; non random sampling.	Composite states have a negative influence on understandability.	No explanation.
15	To determine if UML with an interactive view is more understandable than UML without [71].	Correctness percentage based on number of correct answers; total time in minutes	100 students. Non random sampling.	The total time in minutes to perform the task and a correctness percentage on questions	No explanation.

	Research Goal	Operationalization	Sampling	Observations	Author explanations
16	To determine why subjects found UML difficult [105].	Difficulties perceived by the students	79 students, non random sampling	Training material for UML was perceived to be insufficient; difficult to understand because it is new; inconsistent and confusing diagrams; unclearly defined semantics; hard to memorize UML	i) Theories on cognition and learning explain this. Without proper assistance during the learning process, the learning curve is too steep. ii) Information processing theory explains this too. New knowledge is always mapped to existing knowledge (if there is no compatible knowledge, then this hinders understandability) iii) The UML standardization process led to unclearly defined semantics. iv) Number of constructs to remember exceeds the size of the short term memory of the subjects.

	Research Goal	Operationalization	Sampling	Observations	Author explanations
17	To investigate whether UML models with a higher level of detail are better to understand than UML models with a lower level of detail [91]	The number of correctly answered questions about the models; the time needed to complete the questions	53 students, non random sampling	Higher level of detail reduces the errors made, and questions are answered faster.	Less detail leads to interpretation errors, and therefore to mistakes, and to longer times to figure out the answers to questions.
18	To investigate if UML models with a better layout were easier to understand than models with a bad layout [110].	Correctness of the answers; time used to answer the questions; subject perception.	78 students, non-random sampling	Models with better layout contain less mistakes and are faster to construct. Also subject prefer models with better layout.	No explanation.

	Research Goal	Operationalization	Sampling	Observations	Author explanations
19	To determine if experienced users understand UML models better than inexperienced users [107].	Average accuracy of models that were modified); time spent modifying the models; time spent to identify the part to modify; time spent on a task; total time spent.	21 subjects, students and practitioners. Non random sampling.	Practitioners were more accurate than students; students were faster than practitioners.	The fact that practitioners are more active and face real, industrial design problems daily can justify their ability to perform more accurately than students.
20	To compare five different UML class diagram notations [99].	Correctness of answers; time spent to answer all the questions	39 students. Non random sampling	They observed that intuitive notational variants are better to understand in terms of time spent and correctness of answers	No explanation
21	To compare a RUP UML extension with standard UML on understandability [1].	Time in minutes to understand a model; correctness of answers about the model; perception of the subjects.	39 students. Non random sampling.	The extension was easier to use.	No explanation

	Research Goal	Operationalization	Sampling	Observations	Author explanations
22	To compare understandability of ER with UML models [24]	Correctness percentage of correct answers	3 studies (40) with students and two replications (30,69). Non random sampling.	UML models were easier to understand	UML is more precise than ER, contains less concepts. Therefore easier to understand.

Table 6.2: Overview of GORE understandability experiments (first part) and UML understandability experiments (second part).

7 Third Evaluation [1]

[1]This chapter is based on an article in the proceedings of the Workshop on Conceptual Modeling in Requirements Engineering and Business Analysis [37]

7.1 Introduction

In chapter 3 we have extended the EA modeling language ArchiMate [73] with concepts from goal-oriented requirements engineering (GORE) [29]. The extension is called ARMOR, and the result of extending ArchiMate with ARMOR is called ArchiMate 2.0. So ARMOR is the GORE part of ArchiMate 2.0. This chapter evaluates the understandability of ARMOR.

In chapters 4 and 5 we have investigated the understandability of the ARMOR extension by two case studies [36] and two quasi-experiments with practicing enterprise architects [30]. The results showed that practitioners find ARMOR very complex and use only a few of the concepts of ARMOR correctly. In chapter 6 we investigated possible explanations of why GORE is so hard to understand.

To further test these explanations, we have replicated the experiment with participants of the REFSQ '14 conference that can be considered experts in GORE languages[2]. We additionally asked the subjects for the perceived understandability of ARMOR concepts in an exit survey. The results confirm our earlier findings about understandability problems in goal-oriented notations.

We start with listing the research questions in the section 7.2. Next we describe our research methodology in section 7.3. Section 7.4 describes our conceptual framework. The results from the experiment, the exit survey and the comparison with our previous results are described in section 7.5. Answers to the research questions are summarized in section 7.6.

Section 7.7 describes some implications for practice and further research.

7.2 Research problem

In our courses teaching ARMOR to practitioners we saw that there were understandability issues regarding the concepts. Therefore we started to investigate this problem. This work is a replication of our previous studies. Our research questions are the same as in our previous quasi-experiments from Chapters 4 and 5, extended with two more questions. We added a question to compare subjects' perception of understandability with the understanding they exhibited during the experiment. We also added a fifth question in which we ask about the comparison across all quasi-experiments.

- Q1: How understandable is the ARMOR language?
- Q2: Which concepts are understood correctly and why?

[2]`http://refsq.org/2014/live-experiment/`

Table 7.1: Entry questionnaire.

- What is your highest level of completed education?
- What is your daily function?
- How many years of experience do you have in this function?
- How experienced are you with a (any) requirements modeling notation? (select one: I have no experience / I understand the concepts / I can read diagrams / I can create diagrams / I can teach a requirements modeling technique.)

- Q3: Which concepts are not understood? Why? Does this agree with subjects' perceptions of understandability?
- Q4: What kind of mistakes are made? Why?
- Q5: How much do our findings differ from our previous samples and why?

In all cases, we want to know not only an answer to the journalistic question what is the case, but also the research question why it is the case.

7.3 Research methodology

We performed two identical experiments at REFSQ'14 of 90 minutes each. We could not control any information flow from the first experiment to the second experiment, and we depended on the integrity of the participants, all researchers, to refrain from creating such a flow.

Subjects self-selected into the experiments, and to be able to assess the influence of previous knowledge of GORE concepts, we measured the knowledge and experience of the participants with GORE notations in a short entry-questionnaire (table 7.1).

Each experiment started with a very short lecture (30 minutes) on ARMOR. Next, the participants had to construct simple goal models of a case. They were given 50 minutes for this. To fit within the available time, the difficulty level of this case was very easy compared to the actual real world problems of our previous experiments.

Finally, before leaving the room, each participant filled in an exit questionnaire in which for each of the GORE concepts used in the assignment, it was asked (1) whether they found the concept easy, normal or hard to use, and (2) to optionally explain their answer.

During data analysis, the answers were graded by the first author of the original article in the same way as in the previous experiments. He compared the used concepts to intended use of the concepts and marked if the concepts were used incorrectly. Results were discussed with the second author of the original article[37].

7.4 Defining understandability

In a survey of definitions of understandability of conceptual models, Houy et al. [53] identified five types of definitions: the ability to recall model content, the ability to correctly answer questions about a model,the time needed to answer questions about the model, the ability to solve problems using the model, and the ability to verify a model. These are however measures of model understandability, whereas we are interested in measures of language understandability. An example of a measure of language understandability is the ability of subjects to guess the definition of a language construct by looking at the icons. Caire et al. [19] measured this for i*.

However, these are all measures of passive understanding, whereas we are interested in a more active form of understanding that is closer to the concept of ease of use. How easy is it to construct a model in a language? This concept of understanding is used by, for example, Carvallo & Franch [20] and by Matulevičius & Heymans [79], who measured the number of mistakes made in constructing i* models, and by Abrahao et al., who measured the time needed to build a model [2]. Our concept of understandability is close to the first of these, and we define the understandability of a language construct as *the percentage of users that can use a concept correctly.*

Construct validity is the validity of the operationalizations of a construct. Note that our definition of understandability is close to that of ease of use, and that our results are therefore about a different concept of understandability than that used when studying understandability of a conceptual model. Our definition agrees with that used by other authors [20, 79], but of the two known operationalizations, correctness of use and time to use, we have selected the first one only. This should be taken into consideration when comparing our results with those of others.

7.5 Observations

There were 18 participants in total, about evenly spread over the two experiments. Two subjects had a bachelor's degree, seven had a master's degree and nine a PhD degree. Furthermore, the majority of the subjects considered themselves experts in requirements engineering in either industry or academia. According to the entry survey 9 out of 18 subjects had the ability to teach requirements engineering notations.

Combining this high level of expertise with the relative simplicity of the assignment, we would not expect any serious understandability problems with GORE notations.

Table 7.2: Data about correct construct usage by the 18 participants.

Practitioner	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	avg
Stakeholder	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
Influence	94	100			100	100	100	100	100	0		100	100	100					89
Goal	69	100	100	100	100	100	100	100	100	100	100	100	100	0	100	100	100	100	88
Assessment	100	100	85	100	100	100	40	100	100	100	100	100	100	0	100	100	100	100	82
Realization	0	100		100	100	0	100	100	100	100	100	100	33	100			100	100	78
Requirement	0	86		100	100	20	50	100	100	100	67	100	100	100			100	100	73
Driver	0	100	33	100	100	0	100	100	0	100	100	100	50	100	100	100	100	100	71
Decomposition	33	100		0	50	0	12	29	0	25	0	50		100		100	67	0	19

Table 7.3: Summary of the exit survey.

	Easy	Normal	Hard	Most common explanation
Stakeholder	16	1	1	A very common and well known concept.
Influence	5	7	6	Unknown when to use it.
Goal	6	7	5	Hard to distinguish from driver. Hard to distinguish from requirement. Common concept.
Assessment	7	6	5	Difficult to distinguish from a goal.
Realization	5	5	3	What is a full realization?
Requirement	5	8	5	Very similar to goal. Common concept
Driver	6	9	3	Very difficult to distinguish from a goal.
Decomposition	4	7	7	Unknown when to use it.

Table 7.2 lists the ARMOR constructs on the left and summarizes the scores that the subjects received on their assignments. Row i column j shows the percentage of times that practitioner i used concept j correctly. The numbers are the percentage of correctly used concepts by each subject. When a subject did not use a concept at all, the corresponding cell is empty. The avg column shows the percentage of users that always used the concept correctly. The rows are ordered from best understood to least understood construct. Table 7.3 summarizes the scores of the subjective evaluation of understandability, ordered in the same way as table 7.2. The numbers are the total number of subjects that found a certain concept easy, normal or hard to use. The final column summarizes the most frequently occurring explanations provided by the subjects. We now discuss our findings in detail.

The *stakeholder* concept is based on definitions from TOGAF, i* and Tropos [115, 129, 13]. All subjects used this concept correctly and we conclude that the stakeholder concept is an easy to use concept. This is supported by the subjective evaluation of the exit questionnaire. The explanation the subjects provided is that it is a common concept.

The next best understood construct was that of *influence,* defined in ARMOR as a positive or negative influence of satisfaction of one goal on the satisfaction of another goal. This definition is based the influence concept on i* and Tropos [129, 13]. 89% of the subjects used the influence relation correctly, but only 5 out of 18 users found the relation easy to use. Participants found it difficult to choose between the decomposition and influence relation. The most common mistake was also that it was used instead of a decomposition.

ARMOR defines a *goal* as some end that a stakeholder wants to achieve, a definition common in the GORE literature [119, 129, 16]. 89% of the subjects used the goal concept correctly. The subjective evaluation shows that subjects still had a hard time using the concept. They found it hard to distinguish from the concepts of driver and of requirement. This is consistent with the types of mistakes made as sometimes drivers or requirements were stated as goals.

ARMOR defines an *assessment* as the outcome of the analysis of some stakeholder concern, a definition based on that of BMM [16]. 83% of the subjects used the assessment concept correctly. However, subjects found the concept was too close to a goal. This is supported by the types of mistakes made by the subjects, assessments were confused with goals.

ARMOR defines the *realization* relation as a relation that some end that is realized by some means, a definition found too in i* and KAOS [129, 119]. 79% of the subjects used the realization relation correctly. This is consistent with the subjective evaluation, where only three subjects found it hard to use. The most common mistake was that it was used to relate two requirements.

A requirement is defined as *some end that must be realized by a single component of the architecture* . 69% of the subjects used the requirements concept correctly. The most common mistake was that goals were modeled as requirement. This is consistent with the explanations the subjects provided, that goals and requirements were difficult to distinguish.

A *driver* in ARMOR is that it is a key interest of a user, a definition that is taken from TOGAF [115]. Only 67% of the subjects used the concept correctly, which is consistent with the subjective evaluation. The subjects found it very similar to the concept of a goal. The most common mistake made was indeed that a goal was modeled as a driver.

The ARMOR concept of a *decomposition* is a combination of concepts from the EA and GORE literature [129, 16, 13]. ARMOR defines it as a some intention that is divided into multiple intentions. Only 19% used the decomposition relation correctly. This is consistent with the subjective evaluation where only five users found it easy to use. The subjects found it difficult to choose between decomposition and influence.

Some of the data in table 7.2 are consistent with the subject evaluations of the exit questionnaire. For example, when a subject subjectively found a concept hard to use, often they would not use the concept all. The subjects provided an explanation that the relations were sometimes hard to identify. We believe that therefore they just picked one. This is also the case with the other concepts which were very similar, for example the goal and requirement concept.

There are also discrepancies. For example, 11 subjects found the decomposition relation not hard to use, but only 3 subjects used the relation correctly. Conversely, only 5 users found the influence relation easy to use, but most participants used it correctly. Apparently, perceived understandability does not coincide with understanding.

7.5.1 Comparison With Our Previous Results

The level of understanding exhibited by the participants was much higher than in our previous study with practitioners [30] from chapter 5. In our earlier study from chapter 4, only 5 concepts were used correctly by more than half of the practitioners. This agrees with the higher level of expertise of our current group of participants compared to our previous samples.

However, there is a rough correspondence in the orderings of understandability. In our earlier experiment, the concepts of stakeholder and of realization were used correctly by all practitioners. In our current experiment, the concept of stakeholder was used correctly too, but the concept of realization was used incorrectly by some participants, and they perceived some problems in using it. This may be a consequence of the more academic expertise of the subjects.

In all experiments, the concepts of stakeholder, influence, goal and requirement were the best understood (in that order) and the concept of decomposition was the least understood. And in all experiments, participants had trouble distinguishing requirements, assessments and drivers from goals, and participants wondered why all of these concepts are present in the language.

7.5.2 Explanations

Our observations support the explanations of understandability problems listed earlier. The number of concepts in ARMOR is large, making it difficult for novice users to choose among them. Related to this is the second explanation, which is that the semantic distance among some concepts is very small, making it even harder to choose the right concept to use in a modeling problem.

Finally, the distance of ARMOR concepts and the meaning of those concepts in daily practice is large in our previous experiments. This explained problems that practitioners had with assimilating ARMOR concepts. For the academics that participated in the current experiment, this distance is smaller, because they teach GORE concepts or have studied them. This may explain the higher scores that the participants in the current experiment had compared to the practitioners' score in the previous experiments.

One factor that affects the internal validity of these explanations is that the explanation of ARMOR given by the first author may have created understandability problems. However, The first author regularly teaches these concepts to practitioners. And to prepare for the current experiment, he has explained the concepts to university colleagues. This should mitigate the threat that understandability problems have been caused by the instructor rather than by the language.

7.5.3 Generalizability

Our sample is too small to do any statistical inference. Moreover, the participants self-selected in the sample, which may have biased the results. However, given the fact that our sample consisted of GORE experts who chose to do an assignment with a GORE language, we think that other academic subjects would at least have the understandability problems that we observed in our sample.

We replicated the findings of earlier experiments about most understandable and least understandable concepts, and this supports generalizability too.

Moreover, our explanations in terms of the large number of concepts and the small semantic distance among some concepts, and the need of language users to assimilate new concepts to existing knowledge, are stated in general terms. To the extent that these explanations are generalizable, the phenomena that they explain are generalizable too.

7.6 Answers to research questions

Q1: How understandable is ARMOR? The last column of table 7.2 shows the answer to this. Only the stakeholder concept scored 100% an was perfectly understood. However, the only concept that was not clearly understood was that of the decomposition relation, scoring only 19%. The concepts of driver, assessment and goal were very well understood scoring more than 80%. The concepts of requirement, influence and realization were fairly well understood scoring in the 70% range.

Q2: Which concepts are understood correctly and why? Except for the decomposition relation all concepts were understood (scoring more than 55) This can be explained by that most of the concepts are very common concepts.

Q3: Which concepts are not understood correctly and why? There is a gradation in non-understanding, with the decomposition relationship at the bottom. The decomposition relation is very difficult to distinguish from the influence relation.

Q4: What kind of mistakes are made? Why? Does this agree with subjects' perceptions of understandability? The subjects modeled drivers and assessments as goals, and modeled influence relations by means of decomposition relations Explanations were given above. Apparently perceived understandability does not coincide with actual understandability.

Q5: How much do the results differ and why? The results from this study were roughly similar to the results of our previous work. The major difference is that the subjects scored much better than the subjects in our previous experiments. This can be explained by the higher expertise level of the current subjects, and the greater simplicity of the assignment compared to the modeling task in the previous experiments.

7.7 Discussion

ARMOR is part of an Open Group standard [73] and the concepts we investigated in this chapter will remain present in the language. However, one practical implication of this chapter is that in future training programs we will make a distinction between the recommended minimal concepts such as the concepts of stakeholder, goal, and requirement, and less important concepts, such as those of driver and assessment, that can safely be ignored in practice.

We also have to improve our training material. When we saw that the level of education went up, the number of understandability issues dropped. Somehow we need to compensate a lower level of education or experience with our training material. This can be with practically usable guidelines for the use of the concepts that we do recommend. These guidelines could be tailored to specific experience levels, e.g., develop guidelines for inexperienced participants and different guidelines for experienced participants.

Part III

Business Models

8

Traceability between *e*^{3}*value* and ArchiMate[1]

[1]This chapter is an adaptation of the article in the proceedings of the Workshop on Conceptual Modeling in Requirements Engineering and Business Analysis [27]

8.1 Introduction

In chapters 3-7 we have clarified and evaluated the relation between the business goals of the organization and its EA [29, 36, 30]. However, traceability to business goals captures only part of the motivation for an EA. High-level strategic goals are elaborated in a business model and the business model in turn motivates design choices in the EA [5]. In other words, an EA should not only be used to manage IT costs but also to manage the contribution of IT to the value offerings of an enterprise. In this chapter we extend our work on traceability with a hypothesis about traceability to business models. One could say that in this traceability relationship, the business model provides us with puzzle pieces and the EA will put these pieces together [112].

As the enterprise architecture modeling language we choose ArchiMate [114], both because it is well-known, the defacto standard for EA modeling. We take an ecosystem approach to business models, which means that we see a company as part of a network of companies that coordinate to produce a value proposition. We use e^3*value* as the ecosystem business modeling language [49]. e^3*value* can be used to model the value exchanges in an ecosystem needed to jointly produce a value offering to a customer, and to analyze the long-term commercial viability of this.

In this chapter we show how the EA of one of the enterprises in an ecosystem, specified in ArchiMate, can be aligned with an ecosystem business model, specified in e^3*value*. We expect that realizing this traceability provides us with the following advantages:

- Tracing EA elements not just to the business goals, but actually to quantifiable elements from the business model allows for better reasoning, especially in value networks where goals of different stakeholders need to be brought together to create a shared perception of the ecosystem, and to analyze the commercial viability of the ecosystem.
- To assess which projects that implement the architecture have the most business value in terms of contribution to the enterprise business goals and the ecosystem business model.

In section 8.2 we analyze and reject existing solutions that tried to achieve the same goal and provide an initial analysis. Section 8.3 discusses the ArchiMate strategy layer. In section 8.4 we test the initial hypotheses on a real-world example and, based on this application, refine our initial traceability hypotheses in the form of a meta-model of the hypotheses. In section 8.5 we discuss the validity of our research method and outline some future research.

8.2 Evaluation of Existing Solutions

Before we conduct our initial analysis, we need to decide if we can use existing work. In the related work chapter we briefly introduced and rejected the work by de Kinderen et al. [64, 65].

De Kinderen et al [64] use the business function and the business actor of ArchiMate as the key mapping between a value activity and actor from e^3*value*. We believe the choice for business function is incorrect. We will explain our reasoning. ArchiMate is designed around the service-orientation paradigm [73, 114]. ArchiMate models systems that offer services to the consumers of these services. Even in de case of an organization that offers physical products, services are used to expose the products (i.e., in the case of a car a 'sell car' business service). A business service in ArchiMate should be meaningful from the point of the environment, in this case the users of the business service. The value offered to the user of the service provides the motivation for the existence of the service [114]. ArchiMate also has the clear distinction between externally visible behavior (services) and the internal organization, which is hidden from the environment. In ArchiMate services are exposed to the environment through business interfaces, it is a point of access between the service provider and the service consumer.

In e^3*value* we see these roles as well [49, 44]. e^3*value* offers value propositions between a consumer and a provider in a value network. Essential to this is the notion of a value activity. A value activity is a task performed by an actor that potentially results in a benefit for the actor. A value activity *abstracts away from the internal processes* and stresses the *externally visible outcome* in terms of created value[46]. Associated with a value activity is a value transfer [44] (through the value interface and value ports). A value transfer is the willingness of a provider and a requester to transfer a value object from the first to the second. This means that there are always at least two actors associated, who are economically independent and something of value is exchanged between them. A key component here is the value port. A value port provides the willingness to request and provide value objects from its *environment*. A value port associated with a value activity is also linked to a value port directly connected to the actor through a dependency link to the environment. This way the value adding activities are connected to the external environment.

The business function does not meet these criteria. In ArchiMate [73] a business function describes *internal behavior* performed by a business role. It is an aggregation of behavior based on required business resources, skills, competencies, knowledge, etc. A business function is not directly associated with exchanging value between multiple economically independent actors through some sort of connection to its environment.

The idea of building a bridge and use ArchiMate's layered architecture to operationalize the organization that realizes the value exchanges is something we will use [64]. We agree with the mapping of the actor from e^3*value* to the business actor in ArchiMate. This would represent the organization and can be operationalized using the internal workings of ArchiMate (e.g., adding design decisions).

In follow-up work De Kinderen et al. [65] try to bridge value modeling between ArchiMate and e^3*value* with DEMO [26] based on the transactions from e^3*value* through a complete formal transformation. We reject this design alternative as well. We do not wish to use an intermediate language, because it adds too much complexity for the end user of the languages. Second, this approach also leads to a strange mapping between concepts. We will illustrate this with an example from this work. A transaction from e^3*value* is mapped onto a business interaction in ArchiMate. In e^3*value* a transaction is the set of value transfers triggered when a value interface is triggered [44]. Transactions in e^3*value* involve two or more actors in the value network. Economic transactions are used to determine the profitability of the actors involved. This is not represented by this formal transformation. The resulting transformation, through DEMO, describes the internal process models associated with this transaction. These process models are generated inside DEMO and plotted on the business interaction. However, in ArchiMate a business interaction is, positioned through the meta-model [114], as internal behavior and does not represent the concept of economic transaction as seen in e^3*value*.

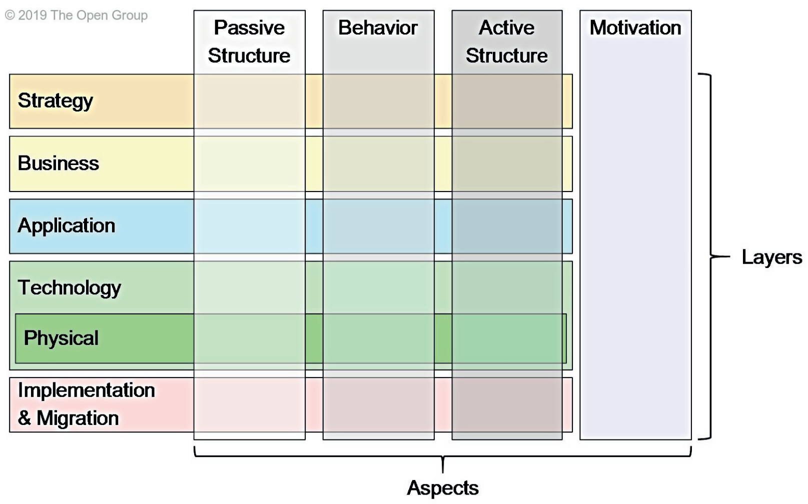


Figure 8.1: The ArchiMate 3.1 framework [73].

8.2.1 Initial Analysis

It is our goal to link e^3*value* with ArchiMate in such a way that ArchiMate can be used to design the organization needed to realize the value exchanges of an organization in the value network, to identify investments and expenses more precisely and collect these for insertion in the e^3*value* model. We also wish to have the ability to reason a about the ability of IT-infrastructure to support the economic transactions from e^3*value*. e^3*value* is designed to investigate the profitability of organizations in a value network. The focus of e^3*value* is to investigate the profitability of each actor in the value network and it lacks an internal operating model and internal organizational design decisions, the focus is on valuable exchanges between the actors in the ecosystem.

ArchiMate [114] is a conceptual modeling language used to describe the enterprise architecture of an organization. In earlier work we co-designed and extensively evaluated goal modeling for ArchiMate. [101, 29, 36, 30]. In the follow-up empirical studies we observed that most of the concepts of ARMOR were conceptually too complex for practical use, and we proposed a simpler version of the language that is more understandable in practice, called ARMOR-light [30]. In ARMOR-light we only use the notions of stakeholder and goals. When a goal is realized by an element from ArchiMate then it is considered a requirement, similar to KAOS [23]. In this chapter we will use

ArchiMate 3.1 as if it only contains the constructs of ARMOR-light.

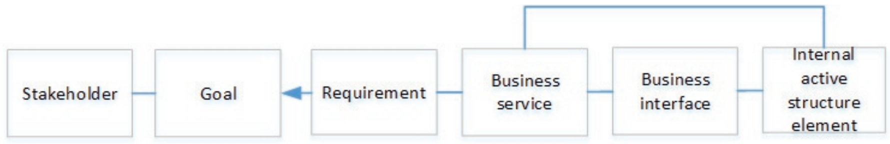


Figure 8.2: Meta-model of ARMOR-Light with part of the metamodel of ArchiMate [30]. The lines represent many to many relations, the arrow represents a subset.

Figure 8.2 shows part of a meta-model of ARMOR-light and ArchiMate. For clarity reasons we have omitted the application and technology layers and a large part of the business layer. Requirements are the subset of goals allocated to a business service. Goals not allocated to an EA element are ends that a stakeholder wishes to achieve. In ArchiMate an internal active structure element is an abstraction of any actor or specialization thereof; e.g. roles, actors, collaborations, etc. A business service is the externally visible behavior of an internal active structure element. It exposes its behavior over a business interface of the internal active structure element, e.g. the sales channel.

ArchiMate models have a business layer, an application layer, and a technology layer, that have traceability links among them. To realize traceability to e^3*value* models, we need to link the business layer of an ArchiMate model to e^3*value* models. Within the business layer, the business service is used to expose behavior and value of the organization to the environment. This is where we expect to find the link between ArchiMate and e^3*value*.

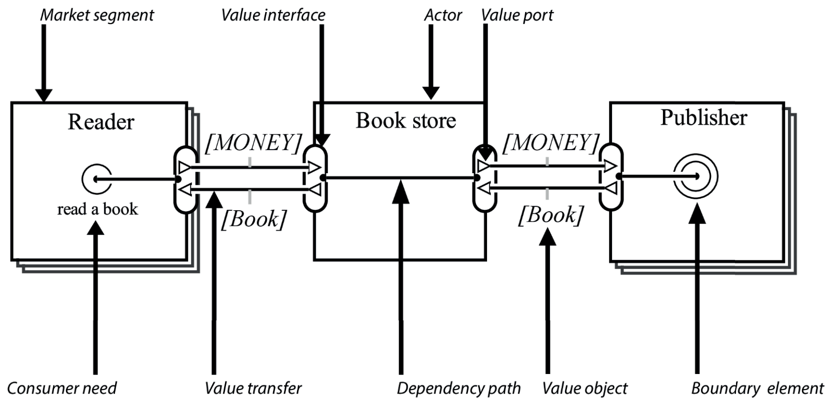


Figure 8.3: Educational e^3*value* model of a bookstore.

In Figure 8.3 an educational e^3*value* model is presented, annotated with the name of the modeling constructs, which we discuss below. In e^3*value* an actor is some entity capable of performing value activities, e.g. a business, department or partner. In the example, the book store is an actor. A special case of an actor is the market segment (e.g. the reader or the publisher). A market segment models that many actors of the same kind.

In e^3*value* this means that all actors in a market segment assign economic value precisely in the same way. A value activity (not shown in the example) is a task performed by an actor which can lead to a positive net cash flow. The value activity differs from activities in process models in e.g. the BPMN. Value activities should be profitable while in BPMN it is perfectly allowed to include activities that only cost money. A value interface represents what the actors offers and requests to/from its environment in terms of value objects.

Value objects are things that are perceived by at least one actor as of economic value. A value interface consists of at least one in-going and one outgoing port, through which the actor requests or offers value objects from or to its environment. The value interface models (1) the notion of economic reciprocity and (2) bundling. Economic reciprocity is the idea that someone only offers something of value, of something else of higher economic value is obtained in return. In the example, the book is exchanged for money, hence the transfers are economically reciprocal. Bundling is the case where it is only possible to offer or obtain value objects in combination. Value ports between actors are connected by means of value transfers, which represent the willingness of actors to exchange things. Internally in an actor, there

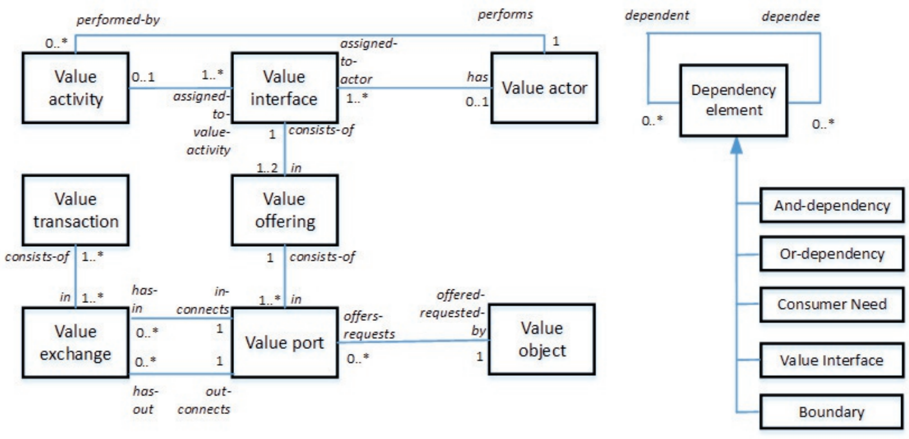


Figure 8.4: Meta-model of actors and dependency paths in e^3*value*[49]

is the dependency path, which shows how value objects are exchanged via a value interface. For example, the sale of a book by the book store requires that this store obtains the book from a publisher. The boundary element of a dependency path indicates the boundary of our modeling interest. Any further transactions that take place in the real world to satisfy the consumer need are not included in our model. Figure 8.4 shows the meta-model for actors (left) and dependency paths (right) in e^3*value* [45]. Based on these two meta-models, we formulate our initial hypothesis about traceability between ArchiMate and e^3*value* concepts in table 8.1 .

8.3 ArchiMate strategy layer

ArchiMate has been extended with a strategy layer that might be relevant for our work. We will start with a description of the strategy layer from the ArchiMate specification [114].

The strategy elements from the strategy layer are typically used to model the strategic direction and choices of an enterprise, as far as the impact on its architecture is concerned. They can be used to express how the enterprise wants to create value for its stakeholders, the capabilities it needs for that, the resources needed to support these capabilities, and how it plans to configure and use these capabilities and resources to achieve its aim.

The strategy layer of ArchiMate operates at a different level than e^3*value*.

Table 8.1: Our initial hypothesis about correspondences between concepts in ArchiMate and in e^3*value*.

ArchiMate	e^3*value*	Argument
Stakeholder	Actor	An e^3*value* actor is always an ArchiMate stakeholder, by definition. The other way around is not guaranteed.
Goal	Customer need	Customer needs are customer goals. Goal models identify and refine customer needs.
Business Actor	Actor	Both are essentially the same thing. An actor in e^3*value* performs activities that produce value. In ArchiMate actors perform behavior as well.
Business Service	Value activity	Both concepts denote externally visible behavior performed by an actor and made available through an interface.
Business interface	Value interface	Both are the interfaces in which the behavior is accessed.

e^3*value* is used to investigate the profitability of an enterprise by focusing on the value transfers between the customers and the organization in a value network. The strategy layer focuses on how an organization can realize value from an internal perspective (similar to an operating model of an organization [93] or the value chain of Porter [98]). The strategy layer answers the question of how value can be realized internally. This is something different than the actual creation of value from the value activities. A value activity always has to lead to a benefit for the actor involved.

When we look inside the strategy layer, we see the following concepts: resource, capability, value stream and course of action. We will discuss them briefly. In essence a course of action is a design decision, or plan to deliver something. This has nothing to do with a value activity. They are strategies and tactics of the operating model internal to the organization.

A *capability* is an ability of an organization. It is something the organization, or departments, can do well. A capability is high-level and aimed at achieving some goal or delivering value by realizing an outcome [114]. A capability is aimed at delivering value, but it is not a value activity. The key differentiator is that a capability is an ability, something an organization can do. A capability influences strategic decision making. A value activity is an activity that leads to a direct benefit through the exchange of valuable objects. In the case of a for-profit organization a value activity should lead to financial profit. A capability can explain how an organization can achieve this.

A *resource* in ArchiMate is an asset owned or controlled by the organization

[114]. Resources can be deployed to realize some sort of competitive advantage. Resources can be used to deliver or realize a value activity, but they are not the same as a value activity.

The closest element possibly related to value activities in the strategy layer in ArchiMate is the *value stream*. A value stream represents a sequence of activities that create an overall result for a customer, stakeholder, or end user. The ArchiMate definition of a value stream is how an organization organizes its activities to realize value for the organization [114]. There is no direct mapping possible between the value stream and a value activity since the activities of a value stream are allowed to be a cost-center. This is not the case with value activities. The end goal of a value activity is that must be profitable. A value stream is also not associated with delivering value to the environment of the organization directly, but indirectly. The value stream in ArchiMate is much closer to a business process, but on a different level of abstraction. A value stream describes what an organization needs to do internally to add value. It very similar in scope and function as the value chain of Porter [98]. In ArchiMate business processes are recommended to realize value streams [114].

We determine that the strategy layer should not be linked with an e^3*value* model directly. The strategy layer is used to operationalize how an organization can deliver value. This is purely internal, from our perspective the strategy layer in ArchiMate is therefore not relevant at the moment. But it can still be used in the way it was intended. Our work can be used in conjunction with the strategy layer of ArchiMate, but it is out of scope of this thesis to discuss this.

8.4 Application: Cirque du Soleil

8.4.1 The example

Cirque du Soleil, based in Montreal, is known internationally for its innovative form of circus production. Cirque was one of the first to reinvent the circus production, without animal usage, but with a focus on artistic human performances [75]. We collected information about this example from public sources on the internet [63, 124] supplemented with some assumptions to round out the example. We will show how Cirque du Soleil offers live shows and virtual shows, offered in an attractive location and through a Virtual Reality (VR) device, respectively. Tickets for the live shows are sold by an independent ticket office. The VR shows are distributed by Samsung. We will construct three different models, since there is no tool yet available to create a single model (see future work). Traceability links are described in the text explaining the models.

8.4.2 Goal model, business model and EA model

We start by constructing a goal model in ArchiMate (figure 8.5). For illustration purposes we have restricted ourselves to one goal per relevant actor.

A goal model like this is often constructed in the strategic phase of EA development, similar to TOGAF's preliminary and vision phases [115]

Figure 8.6 contains the e^3*value* model that illustrates the value adding activities. Value activities are represented by rounded rectangles inside an actor. In our example Samsung enters into a collaboration with Cirque Du Soleil to distribute the VR media of the circus performance to customers. An external ticket office is used to offer a ticketing service. For example, Cirque du Soleil wants to perform a show and Samsung wants to distribute performances.

Customers are represented by two separate actors, Visitor and Digital Customer. Visitors have a need to enjoy a live artistic show, and satisfy this need by paying Cirque du Soleil for performing their value activity. Cirque du Soleil hires a ticket office to sell tickets. The inter-actor transactions and the intra-actor dashed lines form a dependency path in e^3*value* models, connecting a consumer need with all transactions in the ecosystem needed to satisfy the need. The customer need to enjoy a show from home is satisfied by a similar dependency path. Figure 8.7 shows an ArchiMate model of the EA for Cirque du Soleil. We have identified two different main Business Services: the Circus Performance Service and the Digital Distribution Service. These two business services correspond to the value activities of the e^3*value* model from the actors Samsung and Cirque du Soleil. The ArchiMate model also contains four business actors, where Samsung and Cirque du Soleil collaborate together to deliver the digital distribution service. To model the different roles of the customers we have chosen to model the digital customer and the visitor as separate roles.

The same can be seen with the ticket office, they collaborate (the business collaboration) to provide the ticketing service. Since ArchiMate allows for more detail in the modeling of the business services, design decisions like the composition of the ticketing service in the circus performance service are represented here. It is also possible that these translate to supporting internal business services like the recording service. Before you can distribute a show you do need some sort of recording service. This is not necessarily a value adding activity and therefore not visible in the e^3*value* model.

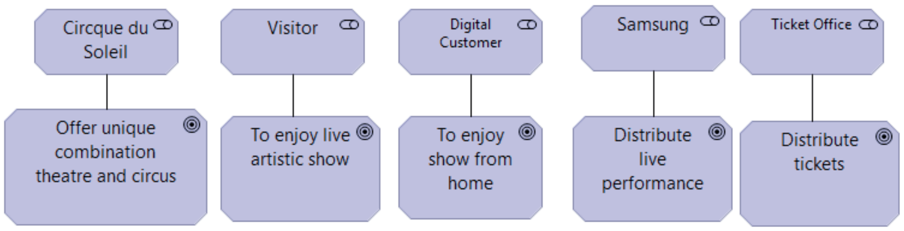


Figure 8.5: Partial Cirque du Soleil Goal Model

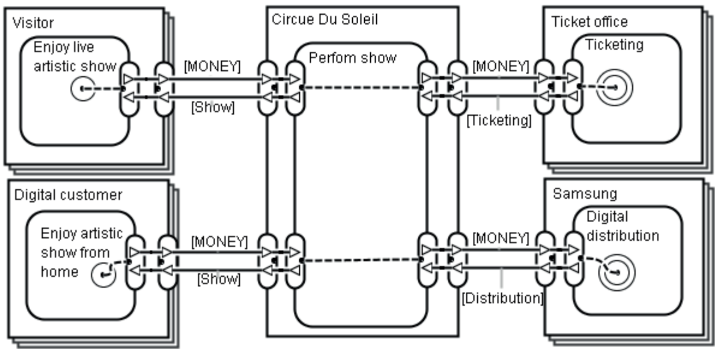


Figure 8.6: $e^3 value$ model of Cirque du Soleil

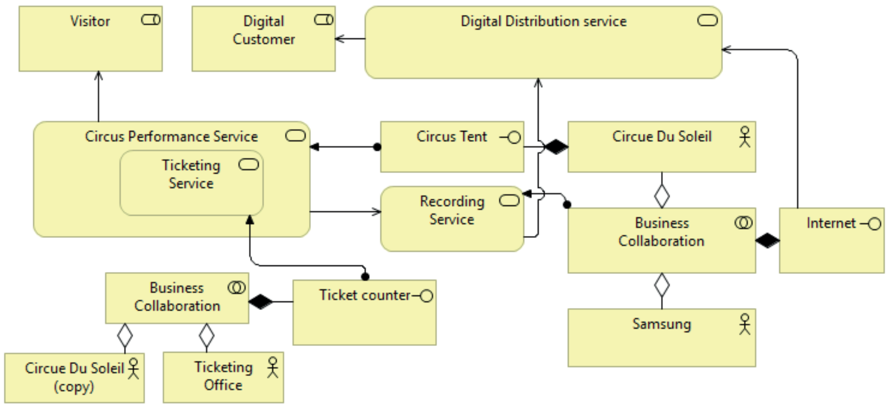


Figure 8.7: ArchiMate model of the business layer of Cirque du Soleil

8.4.3 Observations

Linking ArchiMate goal models to e^3*value* models. First, we see that in our example, goals in the goal model correspond to value activities of business actors in e^3*value*. We believe this to be a general rule for strategic goals. There are goals at every level of the organization, but only strategic goals will be relevant for a business model and may appear there as value activities.

In addition, consumer needs in our example correspond to consumer goals in the stakeholder model. This leads us to the following three refinements of our initial two hypotheses about the correspondence between ArchiMate goals models and e^3*value* business models:

- Stakeholders with strategic goals correspond to actors in an e^3*value* model.
- Value activities in an e^3*value* model correspond to lower level goals in a strategic goal model.
- Consumer needs in an e^3*value* model correspond to lower-level consumer goals in a strategic goal model.

Linking e^3*value* models to ArchiMate EA models. In our example all the value activities that do not have a consumer need attached to them correspond to business services in ArchiMate.

Actors in the e^3*value* model correspond to business actors in the ArchiMate model. This may not be true in general as ArchiMate also contains the concept of a role. An e^3*value* actor may correspond to a role in the ArchiMate EA rather than to a business actor. Future research should provide clarity about this.

The value interfaces in e^3*value* map onto the business interface in ArchiMate. For example, the four value interfaces from Samsung to Cirque du Soleil translate to a single business interface in ArchiMate.

Finally, an e^3*value* dependency path connects transactions among different actors. This may be mapped to business collaborations in an ArchiMate model. Whether this is true in general must be shown by future case studies.

This leads us to the following refinements of our initial hypotheses about the correspondence between ArchiMate EA models and e^3*value* business models:

- e^3*value* actors map to business actors and possibly roles in an ArchiMate EA models.

- e^3*value* activities map to ArchiMate business services.
- e^3*value* value interfaces map to ArchiMate business interfaces
- An e^3*value* dependency path may map to a business collaboration in ArchiMate.

Figure 8.8 summarizes our traceability rules. This meta-model is divided into three different layers, according to the TEAM framework[126]:

- The strategic layer where we find stakeholders and goals,
- The value layer, where we see the value adding activities and
- the technical layer where we find the designs of the organization in an EA.

Integrating e^3*value* into ArchiMate therefore happens at the motivation layer and the business layer. This results in traceability from stakeholder to actor and ArchiMate equivalents. The exact meaning and the cardinalities of the relations are uncertain at this point. We will refine our hypothetical mappings in future chapters of this thesis. We kept the cardinalities and types of relations as conservative as possible as they are only hypotheses.

We hypothesized that a value activity with a nested consumer need can be mapped to a goal. If this is the case, to a maximum of one. The value activity with the nested consumer need is the goal the actor wishes to achieve.

A value activity is mapped to a maximum of one business service and vice versa in a possible traceability relation. But we are uncertain at this point if every value activity can map to a business service or the other way around.

We believe every actor from e^3*value* can be mapped to a single stakeholder. Actors are by definition stakeholders, therefore the relation from actor to stakeholder is 1. From stakeholder and actor we believe the cardinality has be 0..1. Not every stakeholder is an actor, there are many different stakeholders for each organization. We hypothesize that actors from e^3*value* can be linked to either a business actor or a business role (in this model we aggregate these to internal active structure element, as used in ArchiMate). We believe that the mapping should be one to one. The exact cardinalities between a value interface and business interface are unknown at this time, we assume many to many relations.

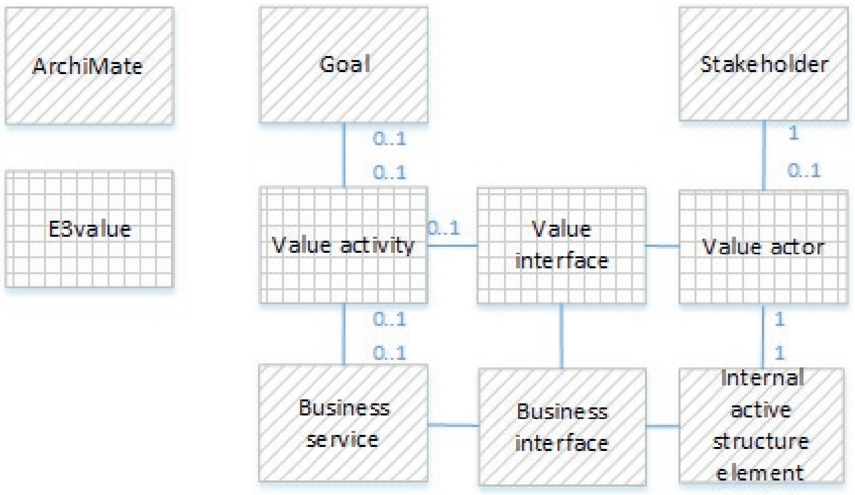


Figure 8.8: Combined traceability model of our hypotheses. All hypothetical relationships are many-many unless otherwise stated.

8.5 Discussion

This work is in its early stages and our current hypotheses are based on an analysis of meta-models plus an application to a single example. We cannot claim generalizability based on this. To test generalizability, we will restrict our research to e^3*value* and ArchiMate. Generalizability to other languages therefore remains an open issue. However, we will investigate generalizability to other cases analyzed in e^3*value* and ArchiMate as a next step.

Within this scope, we need to refine the hypothesis by doing more complex real-world case studies. What is the exact meaning of the relations in our proposed integrated meta-model? What is the relation between value activity and goal? Could it be a realization or specialization? This could also be said for cross-abstraction level of traceability. How do concepts like value and goals relate?

We will test usability of our hypothesis in experiments like we did with ARMOR [36, 30, 29]. Utility in practice will be investigated by means of opinion research, e.g. a focus group of practitioners. A final step is to create a tool-supported method for designing an EA based on a business model, and for extracting a business model from a given EA.

9

Definition of Alignment Guidelines[1]

[1]This chapter is based on the article in the proceedings of the International Enterprise Distributed Object Computing Conference [28]

9.1 Introduction

Earlier, in chapter 8, we have performed an initial investigation of linking e^3*value* models to ArchiMate models [27]. In this chapter we extend on that research.

A BM is a conceptual model of how an enterprise creates, delivers and captures value [93]. Today, business models should represent the value network by which an enterprise collaboratively delivers and captures value [126]. We use e^3*value* as notation to represent business models [49]. As EA notation we will use ArchiMate [114], including the goal modeling extension [37, 36]. One could say that in this traceability relationship, the BM provides us with the reason why an organization exists and the initial puzzle pieces, the EA will put these pieces together [112].

We have three main arguments to combine these topics; first, we want to link the value offerings in a BM to the IT of an organization. This way the realized traceability enables us to reason about the financial benefits of an IT system or project. IT needs operational expenses and investments in IT. There is a clear financial relationship.

Second, a BM only focuses on the value offerings of an organization, but not on the technical and organizational feasibility of the BM. By using EA we can focus on organizational and IT design of the value offerings of an organization to determine the feasibility, possibly with standardized patterns of operationalized business models in ArchiMate. We also believe that by linking the business model of an organization to its EA we can evaluate which organizational components contribute the most to the earnings of an organization. This will help organizations in determining which are the most valuable parts of their enterprise.

Third, if wish to construct an ArchiMate model of a **value network**, as we recommend in previous work [126], we need to know the scope of the organizational network we need to model. Therefore, to be able to design an ArchiMate model for this collaboration of actors, we need to determine the focus of the modeling effort.

This chapter is structured as follows. Section 9.2, where we introduce our research problem. Section 9.3 introduces e^3*value* and ArchiMate. Section 9.4 describes the methodology used. Section 9.5 describes our observations and our new set of guidelines for aligning e^3*value* with ArchiMate. We will illustrate our results with a sample application in section 9.6 and we will conclude with a discussion about validity and future research in section 9.7.

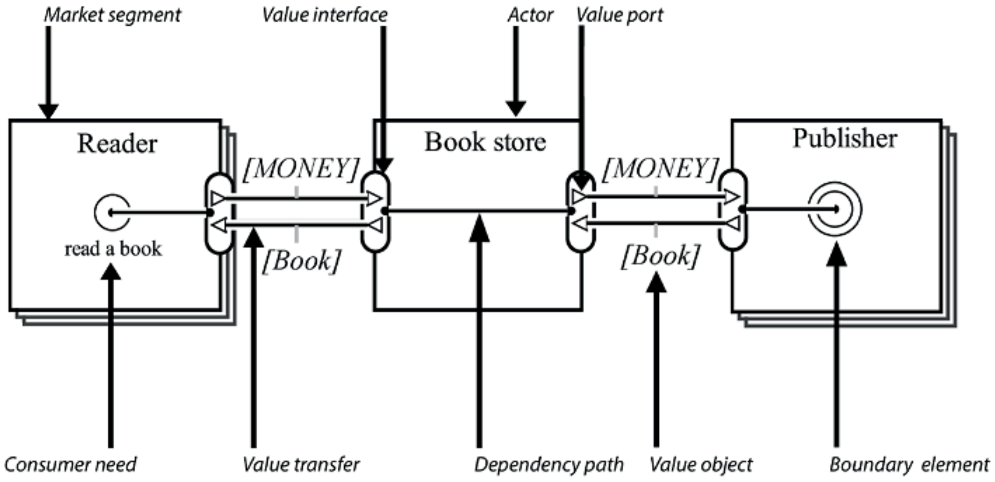


Figure 9.1: Educational example e^3*value*

9.2 Research Problem

The main goal of our research and this experiment is to identify how to align e^3*value* models with ArchiMate models. This task is not straightforward, as both languages operate at a different level of abstraction and were designed with a different goal in mind. e^3*value* focuses on identifying and designing the business services of a networked organization and on analyzing their profitability. ArchiMate focuses on designing the internal organization that realizes the business services. e^3*value* operates on a value network of multiple actors working together, whereas ArchiMate is designed with a single organization in mind.

An organization is an organized group of people with a particular purpose. A value network of multiple actors can also be seen as an organization working together on a particular purpose, namely delivering value to customers. The difference is that the actors in a value network are economically independent, i.e. each must have a positive cash flow. In addition, a value network can contain competitors, who compete for the same customers. The organization represented by a value network is therefore very complex, ranging from a kind of ad-hoc collaboration to a more stable one. This collaboration could have a shared business strategy and shared business goals. Also the designed organization would also use the same concepts as a single organization (e.g.

services, processes, applications, all based on the shared strategy).

First, if we are able to trace from the application to the BM through different ArchiMate layers, we are able to show how an application contributes to the earnings of an organization, which clarifies that IT is not seen as only a cost factor [122]. Second, ArchiMate can be used to model the shared platform needed by the extended organizations in a value network, this would require shared processes, shared IT and a shared infrastructure, all based on the common goals of the organization and strategy. It is our goal to create an alignment between both languages. We wish to develop guidelines to align ArchiMate models with e^3*value* models in such a way that the concepts from e^3*value* have a counterpart in ArchiMate in such a way that we can trace from an e^3*value* model into an ArchiMate model and back.

To address these goals we have conducted an initial conceptual analysis of mapping e^3*value* to ArchiMate business layer diagrams [27]. Only a mapping to the business layer of ArchiMate and the motivation layer is required to align these two languages. If a mapping with the business layer is realized, the other layers will automatically be traceable to the BM. In order to test and refine our initial results, we have organized an experiment with practitioners to answer the following research questions:

- Q1: Which types of mistakes do practitioners make in aligning e^3*value* models into ArchiMate? Why?

- Q2: Which alignment guidelines can we identify based on these mistakes?

- Q3: Which ArchiMate building blocks can we identify based on these guidelines?

Our population of interest consists of conceptual modelers responsible for creating different kinds of organizational models, ranging from value models to EA models. Our goal is not to create formal model-transformation rules that can be automated, in this case a formal transformation is always incomplete. For example, an actor from e^3*value* can be a role or an actor. The e^3*value* language itself has some ambiguity in the concept definitions. Therefore, informal guidelines from e^3*value* to ArchiMate are much more useful as we wish to provide practitioners with understandable tools and guidelines of how to align an e^3*value* model with an ArchiMate model.

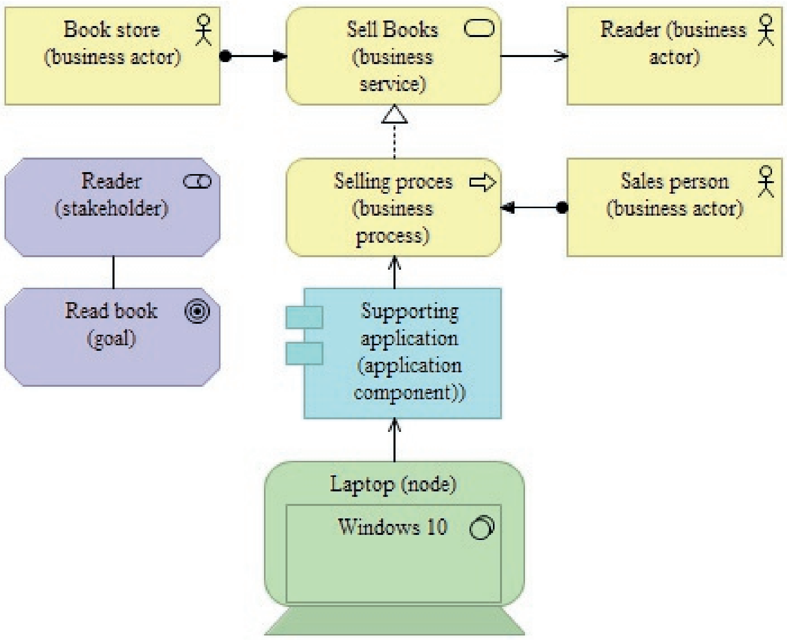

Book store (business actor)
Sell Books (business service)
Reader (business actor)
Reader (stakeholder)
Selling proces (business process)
Sales person (business actor)
Read book (goal)
Supporting application (application component))
Laptop (node)
Windows 10

Figure 9.2: Educational example ArchiMate

9.3 Introduction to e^3*value* and ArchiMate

In figure 9.1 an educational e^3*value* model is presented, annotated with the name of the modeling constructs, which we discuss below. In e^3*value*, an actor is some entity capable of performing value activities, e.g. a business, department or partner. In the example, the book store is an actor.

A market segment (represented by three stacked actors) represents many actors of the same kind. In e^3*value* this means that all actors in a market segment assign economic value precisely in the same way. A value activity (not shown in the example) is a task performed by an actor which can lead to a positive net cash-flow. Value activities differ from activities in process models in e.g. the BPMN. Value activities should be profitable while in BPMN it is perfectly allowed to include activities that only cost money. Also, value activities are much more related to services than to activities.

Value interfaces represent what the actors offers and requests to/from its environment in terms of value objects. Value objects are things that are perceived by at least one actor as of economic value. A value interface consists of at least one in-going and one outgoing port, through which the actor requests or offers value objects from or to its environment. The value interface models (1) the notion of economic reciprocity and (2) bundling. Economic reciprocity is the idea that someone only offers something of value, if something else of higher economic value is obtained in return. In the example, a book is exchanged for money, hence the transfers are economically reciprocal. Bundling is the case where it is only possible to offer, or obtain, value objects in combination.

Value ports between actors are connected by means of value transfers, which represent the willingness of actors to exchange things. The value interfaces of one actor may be related by dependency relations, which shows how value objects exchanged via a value interface require or assume exchanges via other value interfaces of that same actor.

An actor can have a consumer need, represented by a bullet. It may also contain a boundary element, which means that we do not model any further exchanges required to fulfill the consumer need. Starting from a consumer need, we can trace all value interfaces triggered by that need to one or more boundary elements. This is called a dependency path.

For example, the sale of a book by the book store requires that this store obtains the book from a publisher.

In figure 9.2 an educational ArchiMate model is presented. ArchiMate is a language for modeling the architecture of enterprises [114]. This allows organizations to design organizational blueprints based on their business strategy and goals. Figure 9.2 is a possible ArchiMate model based on the e^3*value* model of figure 9.1, using our initial guidelines from table 9.1. We also included a sample application and infrastructure, to show the holistic approach ArchiMate uses.

In this figure we see the central notion of a *sell book*s business service, derived from the book store. We modeled the customer as an end user of the service by using the *serves* relation. The book store is responsible, through the *assignment relation*, for exposing the service to the environment. The reader is also included as a stakeholder with a goal *read book*. The business service is *realized* by a selling process *assigned to* the sales person. An application *serves* the process and the application runs on a laptop with Windows 10 (through the *composition relation*). Please note, that this figure is only for illustration purposes. We omitted most of the concepts and relations of ArchiMate [114].

9.4 Research Methodology

An overview of the initial guidelines from our previous research [27] can be found in table 9.1. We wish to confirm and elaborate on our initial guidelines in an experiment. We did not give our students the initial set of alignment guidelines, as we wanted to test the baseline understanding of the students of aligning these models. That is, we wish to know which concepts or combination of concepts are hard to align to the similar concepts in ArchiMate. This will give us a precise understanding which concepts mappings are not understood and require help.

Our data comes from a group of practitioners who followed a course on creating an EA with TOGAF. The practitioners had no prior experience in e^3*value* so we taught them the basics of e^3*value* modeling. However, they did have prior knowledge of ArchiMate from previous courses and some experience in the field afterwards. The practitioners worked in a broad range of different companies, for example energy providers, telecom providers, the Dutch tax office but also a few smaller sized companies and the defense department. The practitioners all had at least a few years of experience in the organization as a business analyst, functional managers or as administrative employees. At the time of the experiment they were educated at the vocational level, but in their fourth year of their higher vocational education.

This course was part of an evening school for practitioners to obtain their bachelor degree. In the end eight practitioners handed in their assignment, out of 16 students. We disregarded one assignment, because it was of insufficient quality, it did not contain any models to analyze. Seven students passed the course.

We extended this course with value modeling with e^3*value* and asked the practitioners to align their e^3*value* model into an ArchiMate business architecture. We only used the results from the students that passed the course and where they scored sufficiently on the e^3*value* part. The course was supervised and graded by the first author of the original article.

We analyzed the results of their assignment and identified the mistakes made, based on our initial hypotheses set. And based on our analysis we refined guidelines and constructed building blocks of e^3*value* that align into ArchiMate building blocks. Additionally, whilst working with the results from the experiment we were able to perform an additional conceptual analysis. This also led to three new alignment guidelines, given below. The research protocol and the results from analysis are available at request from the first author of the original article. However, the raw data cannot be shared, because of confidentiality reasons. Due to the outbreak of covid-19, combined with the

Table 9.1: Results of our initial analysis [27]

Number	Outcome analysis
H1	ArchiMate stakeholders with strategic goals correspond to actors in an e^3*value* model.
H2	Value activities in an e^3*value* model correspond to lower level goals in a strategic ArchiMate goal model.
H3	Consumer needs in an e^3*value* model correspond to lower-level consumer goals in a strategic ArchiMate goal model.
H4	e^3*value* actors map to business actors and possibly roles in an ArchiMate EA model.
H5	e^3*value* value activities map to ArchiMate business services.
H6	e^3*value* value interfaces map to ArchiMate business interfaces
H7	An e^3*value* dependency path may map to a business collaboration in ArchiMate.

fact the practitioners had to perform their final internship at the same time we were unable to organize closing interviews with the students. We did not plan to do so initially, the option of holding them vanished entirely.

9.5 Results

9.5.1 Observations

We will discuss our observations based on table 9.1, found in our previous work [27], using the same numbering and description. We will provide hypothetical explanations for our observations.

H1. *ArchiMate stakeholders with strategic goals correspond to actors in an* e^{3}value *model.* Most of the practitioners modeled the actors from the e^3*value* model as stakeholders. This can be explained by the fact that an actor is always a stakeholder by definition. This was understood by the practitioners. However, they could not always understand that both languages operate on different detail levels. Instead of modeling the actor as a stakeholder, they modeled the sub-stakeholders part of a composition, combined with the fact that naming was inconsistent throughout the solutions. For example, in one instance the practitioner modeled his/her company as an actor, but modeled this in the ArchiMate model to departments of the company. This can destroy traceability relations. We refined H1 from Table 9.1 into G1 and G4 in table 9.2.

H2. *Value activities in an* e^{3}value *model correspond to lower level goals in a strategic ArchiMate goal model.* No practitioners modeled value activities as a goal. This points at an error in our previous analysis. We identified a possible match between a business goal (problem domain) and a value activity (solution domain). But, these are two different things at a different layer of abstraction. The relationship is more of the means-end type than of an equivalence type relation, were a value activity is the means towards a goal. H2 has been removed.

H3. *Consumer needs in an* e^{3}value *model correspond to lower-level consumer goals in a strategic ArchiMate goal model.* Some of the practitioners modeled consumer needs as goals. However, they did not always indicate they were the same goals. So there were deviations in the names. It was interesting to see that the goals were often part of a larger goal tree. The practitioners did understand that goal modeling resulted in more elaborate models. We refined H3 (table 9.1) into G2 (Table 9.2).

H4. e^{3}value *actors map to business actors and possibly roles in an ArchiMate EA model.* Most practitioners were able to model at least some of the actors from *e^{3}value* as actors in ArchiMate. Roles were not used. The practitioners did make errors in scope, where they forgot to model the main organization and went into detail too fast. This made the models inconsistent. Actors in *e^{3}value* are legal or natural persons, such as organizations or consumers. These can also be actors in ArchiMate, but ArchiMate additionally represents software and hardware entities as actors. Second, *e^{3}value* does not know the concept of a role. This is also caused by that *e^{3}value* is more abstract and generic than ArchiMate. The difference in scope in the two languages is not always well understood. We refined H4 (table 9.1)) into G5 (table 9.2).

H5. e^{3}value *activities map to ArchiMate business services.* Value activities from *e^{3}value* without a consumer need associated to it where mostly modeled as a business service in ArchiMate. The explanation for this is that the definitions of a value activity and business service are very similar, the textbook for *e^{3}value* also refers to value activities as services [49]. Naming was not always consistent and some services were forgotten. The naming errors are the result of not providing clear enough guidelines. We refined H5 (Table 9.1)) into G6 (table 9.2) and G10 table 9.3.

H6. e^{3}value *interfaces map to ArchiMate business interfaces* A single practitioner modeled a sequence of value interfaces on a dependency path as a business interface. However, he did so incorrectly (not adhering to ArchiMate modeling guidelines). Modeling business interfaces is not mandatory in ArchiMate, so many practitioners did not bother to model them. Also, our initial analysis outcome is incomplete. In *e^{3}value*, value activities have value interfaces too, but this cannot be modeled in ArchiMate. We changed H6 (9.1) to G7 and G8 (table 9.2).

H7. *An* e^{3}value *dependency path may map to a business collaboration in ArchiMate.* Not a single practitioner modeled a business collaboration based on a dependency path. However, a single practitioner did model a sequence of business services that was a sequence of value activities in a dependency path.

Apparently it was not clear for practitioners how e^3*value* dependencies and collaborations align with ArchiMate constructs. Perhaps, that the difference in scope between the languages, e^3*value* models a value network instead of a single organization, was something the practitioners found hard to identify. Our initial analysis outcome was incomplete and needs refinement in adding the sequence of business services. H7 from table 9.1 is refined into G11 and G12 in table 9.2.

9.5.2 Design of guidelines and building blocks

Building block 1 Based on these guidelines we identified a number of building blocks. Figure 9.3 illustrates the first building block. We have used the first four guidelines to construct this block. The first four guidelines are all related into aligning the consumer need part of an e^3*value* model to the relevant stakeholder models. Guideline G1 and G2 are used to identify the base concepts of stakeholder and goal. Guideline G3 is applied to construct a goal model using the association relation between the actor and the goal. The fourth guideline is there to correct an incorrect e^3*value* model. We use these four guidelines to identify building block 1.

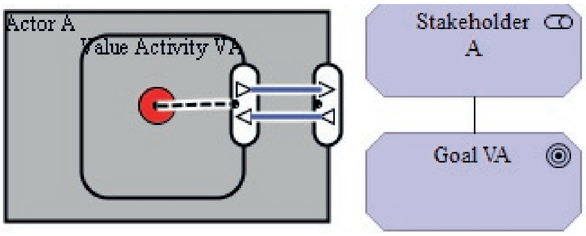


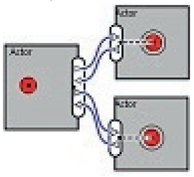
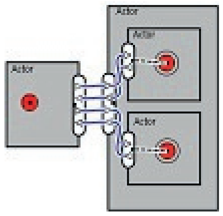
Figure 9.3: Building Block 1, derived from guidelines 1-4

Building block 2 Figure 9.4 illustrates the second building block. We have applied guideline G5, for the identification of the business actor, guideline G6 is applied for the identification of the business service. Guideline G7 combines these two, and guideline G8 identifies the *serves* relation from ArchiMate between the end user and the business service. It is important to note that we decided to follow the guideline associated with guide G7 and model the business interface. The business interface is part of the actor and assigned to the

Table 9.2: Overview of guidelines derived from the experiment

No	H	Guideline	Additional advice
G1	H1	An e^3*value* actor or a market segment can be included as a stakeholder in the ArchiMate motivation layer with the same name.	Additional detail can be added to the stakeholder using the composition or the aggregation relation in ArchiMate.
G2	H3	A consumer need can be modeled as a goal from the ArchiMate motivation layer with the same name as the value activity from e^3*value*.	Construct a complete and correct goal model if needed.
G3	H1 H3	When the e^3*value* actor has a nested value activity and contains a consumer need symbol, this combination can be modeled using stakeholder, goal and association relationship from the motivation layer of ArchiMate.	This is a combination of G1 and G2.
G4	H1	When a e^3*value* actor has a need directly associated with it, the actual need is unknown, therefore we only model the stakeholder with the same name.	Complete the goal model from the stakeholder to create a correct goal model.
G5	H4	An e^3*value* actor can be at least an ArchiMate business actor with the same name.	ArchiMate business actors can be internal or external to the organization itself. In ArchiMate we can identify additional business actors that are part of the same organization. Decompose the business actors when needed.
G6	H5	A value activity, without containing a consumer need, can be modeled as a business service in ArchiMate from the business layer with the same name.	Services can be internal or external to the organization itself. Focus on the organization that is being modeled.
G7	H6	When a value activity is nested in an e^3*value* actor, this may be modeled in ArchiMate by a business actor, business service and business interface. ArchiMate guidelines can be followed for the relations between them.	As an alternative to business actors, business roles can be used instead of the business actor, if the role is responsible for delivering the business service. See G10.
G8		If an e^3*value* actor contains a consumer need and is linked to a value activity as described in G7, this can be an end user of the business service in ArchiMate and then a serving relation is used between the business actor and the business service.	
G9	H7	When there is a trace through the dependency path, including and/or dependencies, this trace must be represented as well in ArchiMate through the business services if the trace contains value activities.	There are different relations in ArchiMate a) triggering, b) information flow, c) serves relation, d) a combination. Use ArchiMate guidelines to identify the correct one.

Table 9.3: Overview of guidelines derived from additional conceptual analysis

No	H	Guidelines	Additional advice
G10	H6	If there are different types of a single actor in e^3*value* which are responsible for performing different kinds of behavior, then consider a business role in ArchiMate. Value activities in an e^3*value* actor can lead to different roles of an ArchiMate business actor.	Actors in e^3*value* often are roles.
G11	H7	If two e^3*value* actors address a single value interface to address a consumer need, then use a business collaboration. 	Investigate the dependency path to determine if we have a collaboration of multiple companies. Name the actors accordingly to the e^3*value* model. Name the business collaboration using the ArchiMate standards.
G12	H7	If two actors bundle their interfaces in a partnership, then a business collaboration and a specialized business service combined with a business collaboration can  be used.	

business service. Guideline G8 in this case is not broken, because the derived relation is still an assignment relation. If the business interface is not used, then the actor is directly assigned to the service. Guideline G11 can applied to adapt this building block. Two actors address a single value interface, and corresponding consumer need, with services of their own. The building block can be extended with these additional services and actors.

Building block 3 The third building block is illustrated in figure 9.5. We have used guideline G9 the design of this block. The focus on this block is the relations that are possible between the value activities and business services. For this reason we have omitted all irrelevant constructs in the ArchiMate model to make this clear. Guideline G9 aligns a sequence of value activities in

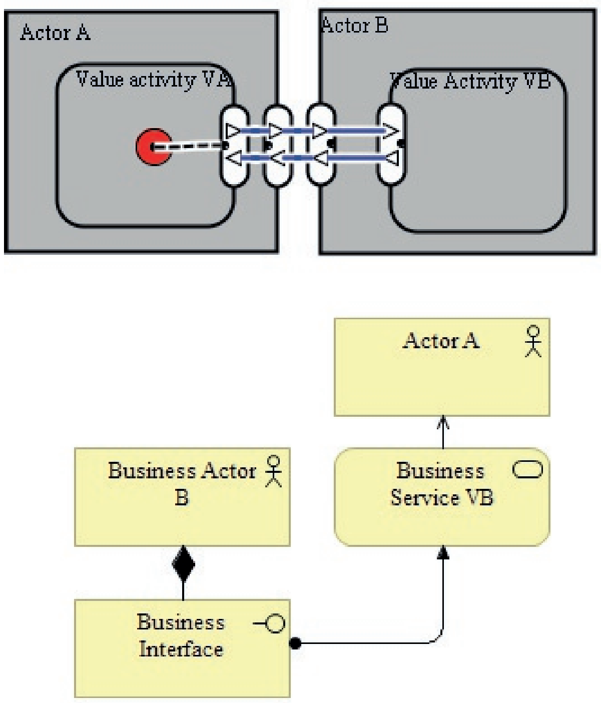


Figure 9.4: Building Block 2, derived from guidelines 5-8

a chain of business services in ArchiMate.

Building block 4 Building block 4 is illustrated in figure 9.6. For clarity reasons we have omitted the actual relations and variations of the business service, but the label of the business service mentions that it should be a specialized or composed service. Two actors and the partnership from e^3*value* work together in a business collaboration. This business collaboration delivers a specialized service. Instead of two companies delivering two services to a single customer, they combine their services. Guideline G12 is used for this. If there are more actors involved in this partnership, the building block has to be extended with this.

Guideline G10 is used to map an actor into roles in ArchiMate. This is an alternative for guideline G1. We will not include a building block for this,

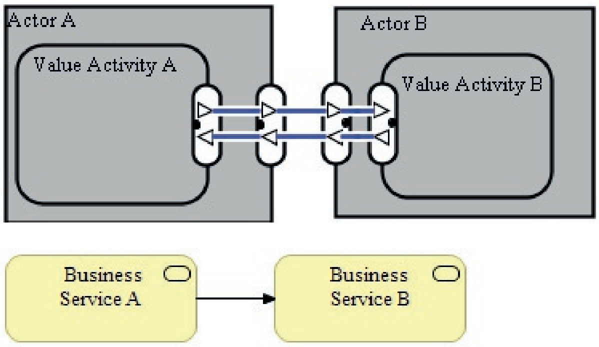


Figure 9.5: Building Block 3, derived from guidelines 9

since this is not an aggregate guideline set.

9.5.3 Answers to research questions

Q1: Which type of mistakes did the practitioners make in aligning e^{3}value *models with ArchiMate? Why?* The practitioners made in general two types of mistakes, which were probably caused by the different scope of both languages and the complexity of a value network. For example, practitioners forgot to model a high level actor as a stakeholder, but did model elements that could be part of a stakeholder decomposition, without the top stakeholder. Second, aligning a value network (e.g. a chain of services) was also hard for the practitioners.

Q2: Which alignment guidelines can we identify based on these mistakes? We were able to identify 12 guidelines based on the experiment and the following additional conceptual analysis. Table 9.2 and table 9.3 answer this research question.

Q3: Which ArchiMate building blocks can we identify based on these guidelines? We were able to identify four building blocks, illustrated in figure 9.3 through figure 9.6. These figures answer our third research question.

9.6 Application

We illustrate our guidelines by applying them on our example case of Cirque du Soleil. Figure 9.7 illustrates our *e^{3}value* model and figure 9.8 our resulting

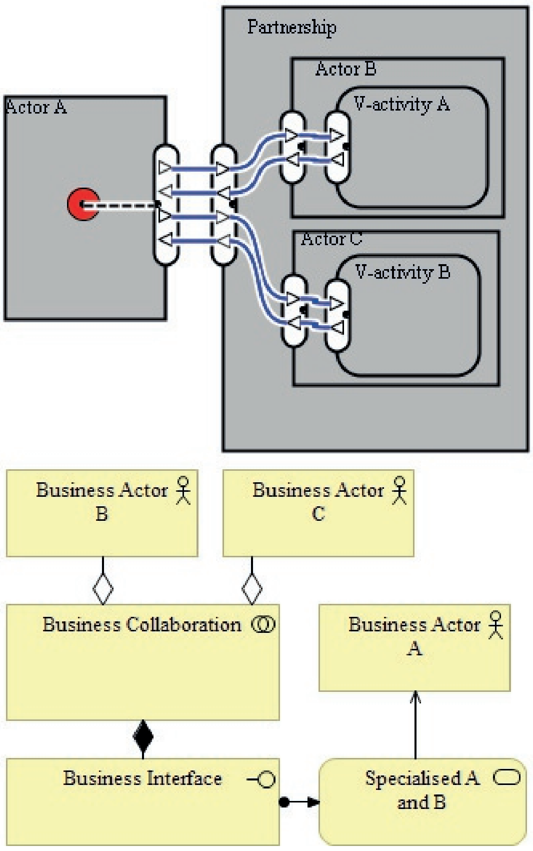


Figure 9.6: Building Block 4, derived from guidelines 11 and 12

ArchiMate model. The central actor here is Cirque Du Soleil, their value activity is to perform a live show. Visitors have a consumer need to enjoy a live artistic show and the consumer satisfies this need by paying Cirque du Soleil for performing. Cirque du Soleil hires a ticket office to sell tickets. Samsung enters into a possible collaboration with Cirque Du Soleil to distribute the VR media of the circus performance to customers. An external ticket office is used to offer a ticketing service.

In our example we modeled a single customer with two value activities.

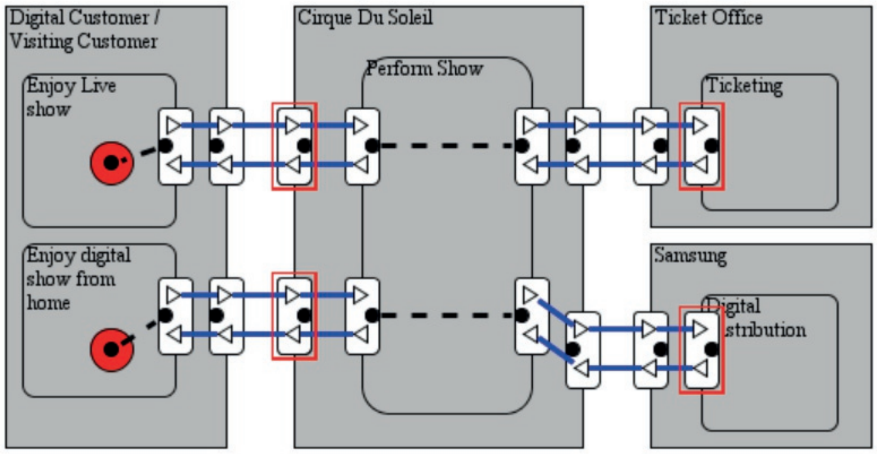


Figure 9.7: e^3*value* model Cirque Du Soleil

The customer and the value activities map to the stakeholder customer with a goal enjoying the live show. This follows guidelines G1, G2 and G3. The customer is also mapped to a business actor, therefore applying guideline G5. Guidelines G6, G7, G8 and G9 are also used in the example. Cirque Du Soleil has a value activity to perform a show. We can therefore identify a similarly named business service in ArchiMate. Since the value activity is nested, we also need to show this in ArchiMate. This is done by modeling the actor Cirque Du Soleil and (indirectly) *assigning* this to the service. We have chosen to include a business interface. We applied guidelines G6, G7, G8, G9. The value activity of Cirque Du Soleil is linked to the visitor. Which is an external stakeholder with a consumer need. Therefore in ArchiMate we model this as a *serves* relation between the business service and the external actor. If we investigate the value activities in the actor visitor, we can also apply guideline G10.

The two value activities denote two possible roles the actor can perform (visiting or watching from home). This is illustrated by G10. Guideline 11 is not applicable here, because the Ticket Office and Samsung deliver to Cirque Du Soleil and not directly the consumer need. These services are supporting for the main service. We have illustrated three building blocks in figure 9.8. Building Block 1 is the, simplified, goal model. Building Blocks 2 and 3 illustrate the modeling patterns we identified by applying the rules.

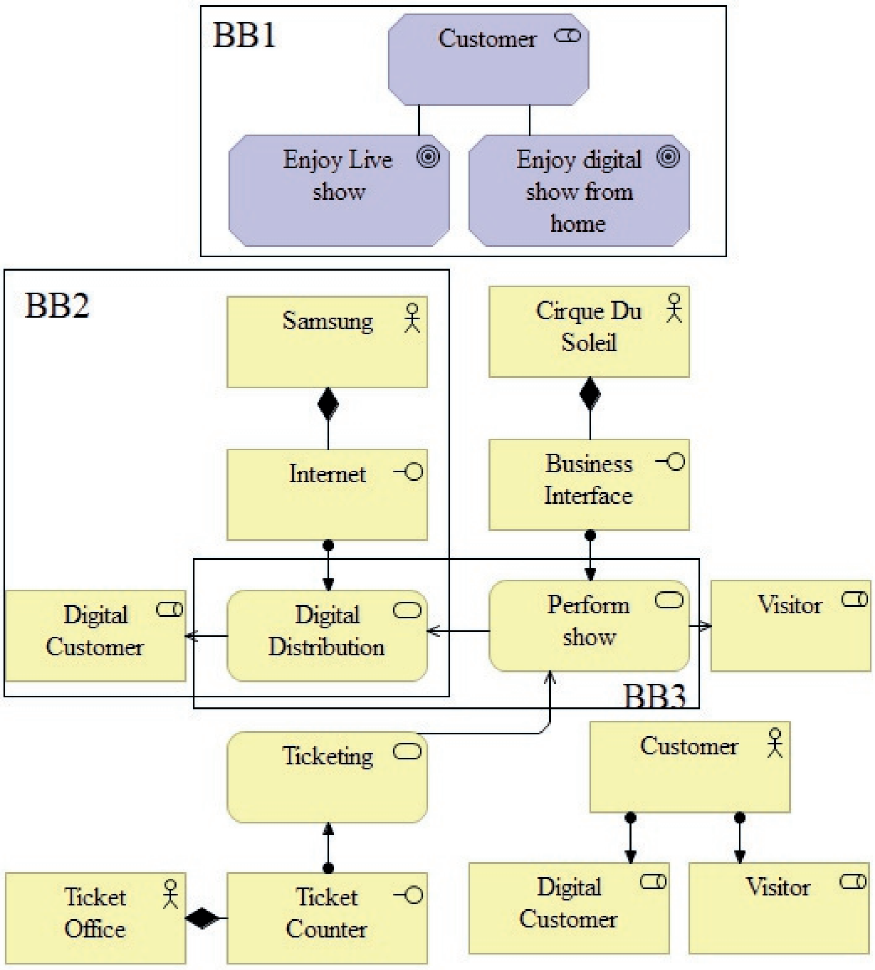


Figure 9.8: Resulting ArchiMate model

Traceability is realized through the mapping of the business service in ArchiMate to the value activity of e^3*value*. This way we can trace the relations of the infrastructure, application and business layers into the business model. This enables reasoning about the contribution of IT to the BM of Cirque Du Soleil and shows what IT architecture is needed to realize the BM. This would require finishing the EA model into the application and infrastructure layers, which we have chosen to omit in this chapter.

9.7 Discussion

9.7.1 Validity

Internal validity is the support for our hypothetical explanations of the phenomena. Could subjects have misunderstood some concepts for other reasons than the ones we hypothesize? Because our previous work is published, we can not guarantee they did not use our previous alignment guidelines. However, the first author of the original article did ask during the last lecture if any practitioners used our previous results and the answer was negative.

External validity is the support for generalization from our quasi-experiment. Generalization to other languages is not our goal. Their homework was an exercise based on an actual problem in their organization. These were real design problems in the organization and therefore a fair representation of the difficulty level. However, this does not mean that we can generalize to all realistic problems. We created a first refinement of our guidelines, future applications might introduce additional refinements (i.e. other problems from practice can introduce different models with different guidelines).

The participants of the group self-selected into the course and so they may be more motivated or more talented than the 'average' business analyst. They were also highly motivated to pass this course, since this was the last course before they could start their bachelor thesis assignment and their company paid for this course. Not passing would reflect badly on the practitioners and would weaken their position in their organization. Based on this generalization to other practitioners is not possible.

However, using this experiment we could identify points of improvement for our hypotheses. We motivated every resulting guideline in terms of the semantics of e^3*value* and ArchiMate, and this motivation ensures applicability to other cases in which these two languages are applied.

A second issue might be that the students constructed an EA based on their knowledge of the actual problem instead of the e^3*value* business model.

We do not believe that this is the case, since the e^3*value* and ArchiMate models were quite similar.

9.7.2 Applicability

We envision a number of different applications for this EA designing approach. First, we can relate, through the business service to value activity mapping, the IT systems of an organization to the earnings of an organization. This would allow us to provide qualitative reasoning about the contribution of IT to the financials of an organization. Quantitative reasoning is still beyond the scope of the current guidelines. It is possible to follow the relations from IT to the value offerings, but there are different kinds of relations. We first would need to determine how to treat these different kind of relations in an ArchiMate model.

Second, a good BM has to be practically feasible and economically viable. Traceability to an EA will help assess both feasibility and viability. In addition, organizations need to start thinking of aligning their EA with a shared EA based on the entire value network. Our technique would allow us to design the shared aspects of this collaboration.

Third, we can operationalize a BM of an organization into ArchiMate business layer diagrams. If we apply this often enough we can derive organizational patterns that operationalize business models. This would be an addition to the work of reference architectures, but instead from a bottom up perspective we can derive them from a top down manner.

9.7.3 Limitations

There are a few clear limitations of this work. Although we have derived our alignment guidelines from a set of different business models, we still need to apply them in practice. No practitioners have used our guidelines. The aspect of business value to the realized traceability is not yet evaluated either. However, from a conceptual analysis point of view, this is now technically possibly due to the mapping between value activities and business services.

There is also no methodological support to align both phases of business model design and EA design. It is important that both disciplines are aligned together, similarly as TOGAF does in their ADM for EA design, EA governance and project management [115]. We also plan to organize small research projects where students will apply these rules on existing e^3*value* models and derive business service architectures from this. We wish to investigate these to identify refinements to our building blocks. We wish to investigate the variations that are possible to determine a more robust set of guidelines and building blocks. We also plan to extend this work with designing a governance organization that steers the value network into achieving its shared goals [126]. We need to incorporate some peer to peer governance techniques for this [62]. And finally we wish to develop a tool where both e^3*value* models and ArchiMate models can be linked together and maintained. This tool would also enable us to implement the different forms of analysis we envision by combining e^3*value* and ArchiMate.

10

Evaluation of Alignment Guidelines[1]

[1] This chapter is based on an article in the proceedings of the Exploring Modeling Methods for Systems Analysis and Development Conference [31]

10.1 Introduction

Commercial services and physical products rely heavily on ICT. For example, Netflix and Spotify would not have been possible without the large scale deployment of content servers and networks. Physical products often have digital twins, which complement the product with additional features, allowing for simulation, training, etc. Since ICT is an intrinsic part of the value proposition of an organization, it can not be considered as a cost-only factor. ICT should be part of value proposition design.

Additionally, many products and services are offered as bundles in complex *business ecosystems*, where each enterprise focuses on its core competence and jointly they satisfy a complex customer need. Following Moore [89], we define an ecosystem as a “collection of companies that work cooperatively and competitively to satisfy customer needs.”

In order to be financially sustainable, an ecosystem requires a *business value model* (henceforth called “business model”), which we define as a conceptual model that represents the creation, distribution, and consumption of economic valuable objects in a *network* of participants, namely the ecosystem [44]. Valuable objects are the outcome of services and physical products that satisfy customer needs, as well as payment for these; also called the reciprocal value transfers.

With ICT-intensive services and products, the design of the provisioning *Enterprise Architecture* (EA) is part of business design. An EA is a high-level conceptual model of an enterprise designed to put the business strategy of an organization into operation [131]. Ideally, in case of ICT intensive products and services the EA puts the business model into operation and hence contributes directly to the profit of enterprise. For this, we need an approach to design the EA of ICT-intensive products and service *in concert* with the business model of the ecosystem. Currently, there is no such approach.

As we take a network view of business models, we use e^3*value* as the business model notation [49, 44]. In accordance with our networked view, EAs too should be extended to an ecosystem of enterprises [126]. We use ArchiMate [114] as the EA language, where we focus on its capability to model business services and collaborations.

An e^3*value* model focuses on actors in a value network and the economic feasibility of the value adding activities in an organization. An ArchiMate model operationalizes this in terms of business services and collaborations, business processes and applications needed to realize this. The models contain different, but also partly overlapping information. An alignment from an e^3*value* model to an ArchiMate model can therefore not be automatic and the guidelines defined in this chapter must be complemented with design choices. The contribution of this chapter is that we provide a real-world validation of the guidelines we developed earlier, and present further improvements to the guidelines to make them more precise. We also elicit an evaluation from practitioners of how to use the conjunction of e^3*value* design and EA design for investment analysis. This chapter is structured as follows. In section 10.2 we describe our research methodology and research questions. Section 10.3 lists our redesigned guidelines and section 10.4 describes our case study. We discuss our results in section 10.5.

10.2 Methodology and research questions

We have developed guidelines for business-model-driven EA design in three iterations of the design cycle [127].

- **Conceptual design:** We analyzed the meta-models of e^3*value* and ArchiMate to define an initial version of the guidelines (version 1). We tested it on a small real-world example: an EA for the Cirque du Soleil [27], discussed in Chapter 8.
- **Lab validation and redesign:** We refined the guidelines in an experiment where we compared the EAs designed by practicing architects from a business model in a laboratory assignment, with the EA that results from our application of the guidelines [28], discussed in chapter 9. Although the assignment took place in the lab, the cases for which the architects designed an e^3*value* model and an EA were from the real world: the companies where they were employed. Analysis of the experiment led to a redesign of the guidelines (version 2).
- **Real-world validation and redesign:** We applied the guidelines to a real-world case to redesign the business layer of the enterprise architecture of an enterprise. This is the case study reported in this chapter. This experience led to a further improvement of the guidelines (version 3).

Although version 3 of the guidelines are the result of applying version 2 on the case study, for readability we present version 3 in section 10.3 before presenting the case study in section 10.4.

The case study is technical action research, as it has two goals: to learn more about the guidelines and refine them, and if we can construct a correct business layer architecture from a e^3*value* business model diagram [127]. We validated the resulting enterprise architecture, and the e^3*value* model that we designed, with management and the enterprise architect of the company. This allowed us to answer the following research questions:

Q1 Do the guidelines produce a correct enterprise architecture of the business model?

Q2 Is the resulting traceability relation useful to make investment and divestment decisions?

To preserve confidentiality, we refer to this company as company X and we changed some details in the models that we will present.

10.3 Redesigned guidelines

Figure 10.1 provides the legends of e^3*value* and ArchiMate that we use in this chapter [2]. Tables 10.1, 10.2, 10.3 present some of our revised guidelines. Table 10.1 gives one of the guidelines to align an e^3*value* model with an ArchiMate motivational model [3]. Basically, the idea is that:

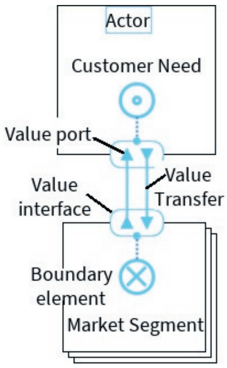
- An e^3*value* actor is mapped to an ArchiMate stakeholder. The reason is that an e^3*value* actor is an actor who has something to gain or lose.
- A need of an e^3*value* actor is mapped to an ArchiMate stakeholder goal. The reason is that a need in e^3*value* is a lack of something valuable that the actors wants to acquire. In other words, it is a goal.

Tables 10.2 and 10.3 list the guidelines for designing an ArchiMate business layer model from an e^3*value* value model. Compared to version 2 [28], we improved the mapping of ports and collaborations, merged a few rules and added rules for and- and or-gates. The tables show the guidelines for designing an EA of a focal company, which is embedded in a network of companies. To design the EA of more than one company, the guidelines have to be applied to each company.

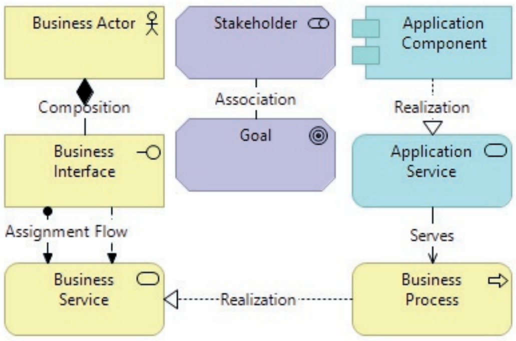
- An e^3*value* actor is mapped to an ArchiMate business actor.

[2]See the e^3*value* user guide at `https://e3value-user-manual.thevalueengineers.nl/` and the ArchiMate documentation at `https://pubs.opengroup.org/architecture/archimate3-doc/`.

[3]A version of the chapter with the complete table is available at `https://www.thevalueengineers.nl/pdf/EMMSAD-2021-long.pdf`.



(a) Legend for e^3*value*.



(b) Legend for ArchiMate.

Figure 10.1: Legends for e^3*value* and ArchiMate

A e^3*value* actor is an entity that is responsible for its survival and well-being, e.g. a profit-and-loss responsible company or a consumer [44]. An ArchiMate business actor is a business entity that is capable of performing behavior [114]. This implies that all e^3*value* actors are ArchiMate actors but not the other way around. By definition an ArchiMate actor is always a stakeholder, as he has something to gain or lose. e^3*value* actors can therefore be depicted on both the stakeholder and business actor concepts.

- An e^3*value* value activity of an actor becomes an ArchiMate business service of that actor.

  An e^3*value* activity is a task performed by an actor that potentially results in a benefit for the actor [44]. An ArchiMate business service is explicitly defined behavior that a business role, business actor, or business collaboration exposes to its environment [114]. We view every e^3*value* activity as a potential business service exposed by the focal company. A value activity is connected to its environment through a value port, similarly as a business service is connected to its environment. This way, a value activity cannot be mapped to a business process, since a business process is internal to the organization.

- An e^3*value* value port of an actor becomes an ArchiMate business inter-

Table 10.1: e^3*value* mapping to the motivation layer of ArchiMate.

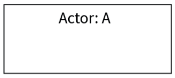
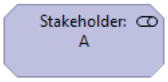
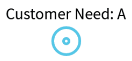
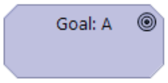
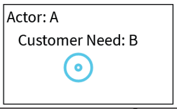
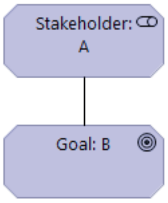
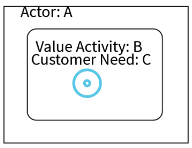
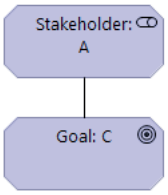
No	Guideline	Additional advice
G1	An e^3*value* actor or a market segment can always be modeled as a stakeholder in the ArchiMate motivation layer with the same name. By definition an actor is always a stakeholder. This is not true for the other way around. 	Additional detail can be added to the stakeholder using the composition, aggregation or specialization relation in ArchiMate. It is not always necessary to model every actor as a stakeholder. This is a choice the enterprise architect has to make. 
G2	A customer need must be modeled as a goal from the ArchiMate motivation layer with the same name as from e^3*value* 	Construct a complete and correct goal model if needed. 
G3	When the e^3*value* actor contains a customer need, this combination must be modeled using stakeholder, goal and association relationship from the motivation layer of ArchiMate. 	This is a combination of G1 and G2. 
G4	When the e^3*value* actor contains a value activity and a customer need, this combination must be modeled using stakeholder, goal and association relationship from the motivation layer of ArchiMate. 	This is a combination of G1 and G2. The value activity is not mapped into the motivation layer. 

Table 10.2: e^3*value* mapping to the Business layer.

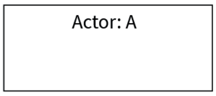
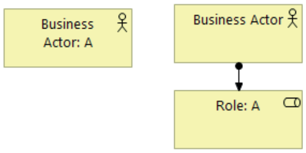
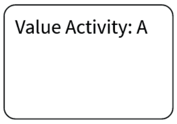
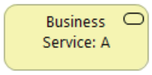
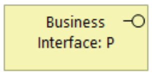
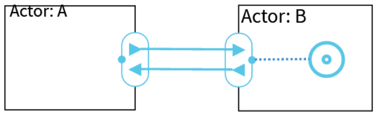
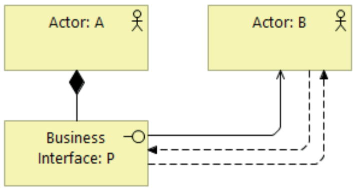
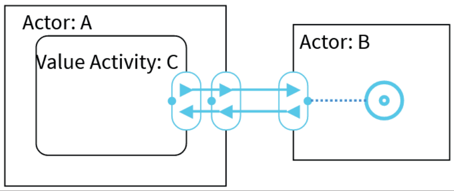
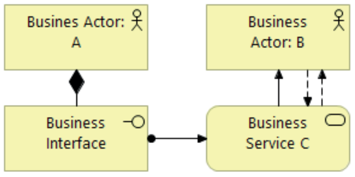
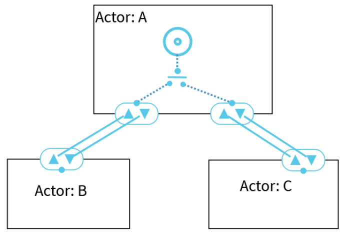
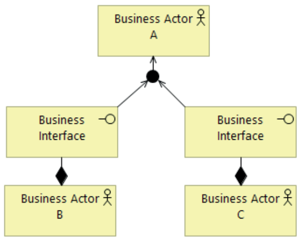
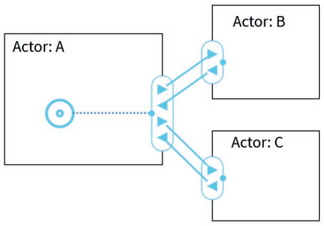
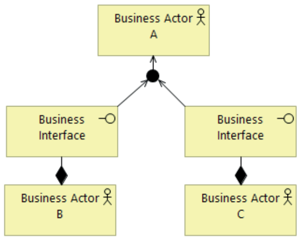
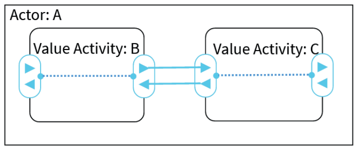
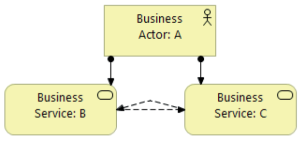
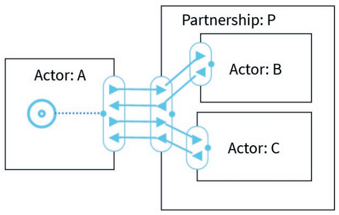
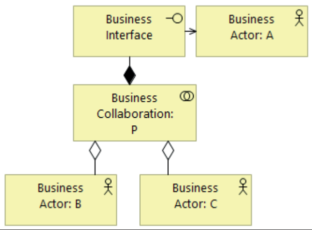
No	Guideline	Additional advice
G5	An e^3*value* actor or market segment is mapped to an ArchiMate business actor with the same name, or to an ArchiMate actor that assumes a role with that name. . 	In ArchiMate we can identify additional business actors. For example, we may identify actors internal to an organization and we may decompose an actor. 
G6	An e^3*value* value activity is mapped to an ArchiMate business service with the same name. 	Services can be internal or external to the organization itself. Additional detail and service composition might be required. 
G7	e^3*value* value ports are mapped to ArchiMate business interfaces	One or more value ports from one or more value interfaces of a same value activity can be mapped to the same ArchiMate business interface. 
G8	An e^3*value* value exchange is mapped to ArchiMate flows. In addition, the exchange can be mapped to an ArchiMate serving relation in the direction of supplier to customer. Ports of the focal company will be mapped to one or more ArchiMate business interfaces. 	If B contains a boundary element instead of a need, the direction of the serving relation would be reversed. 
G9	An e^3*value* activity connected through a value exchange to a need of an actor is mapped to an ArchiMate business service serving the actor. 	If B contains a boundary element instead of a need, the direction of the serving relation would be reversed.  Busines Actor: A Business Actor: B Business Interface Business Service C

Table 10.3: e^3*value* mapping to the Business layer

No	Guideline	Additional advice
G10	An AND/OR gate in e^3*value* maps to an and/or junction in ArchiMate. 	For the or junction the connector in ArchiMate is a hollow circle. Flows can be added as needed as in G8. A junction in ArchiMate connects relations of the same type.  Business Actor A Business Interface Business Interface Business Actor B Business Actor C
G11	Two e^3*value* actors connect to a single value interface to address a customer need are mapped to an and-junction in ArchiMate in the same way as in G11. 	 Business Actor A Business Interface Business Interface Business Actor B Business Actor C
G12	An e^3*value* value exchange between two value activities inside an actor maps to mutual flows between two ArchiMate business services of that actor. 	The flow relation denotes the transfer of money, information or goods. If the direction of the dependency path is known, this can be represented by a serving relation in ArchiMate.  Business Actor: A Business Service: B Business Service: C
G13	A composite actor in e^3*value* is mapped to a business collaboration in ArchiMate. 	The business collaboration will offer services from both actors as a bundle. Use a composition or aggregation relation between the parent and child services.  Business Interface Business Actor: A Business Collaboration: P Business Actor: B Business Actor: C

face of a that actor. This is a change of version 2.

> An e^3*value* value port is a willingness to provide or request something of value (a value object) [44]. ArchiMate business interfaces are channels through which a business service is made available. An value port has no direct counterpart in ArchiMate. However a value port can be composed into an ArchiMate business interface. This adds a channel. Value ports from multiple value interfaces from an OR dependency graph can be mapped to a single business interface. One or more value ports of a single business interface or an AND dependency graph can also be represented as a single business interface. When there are different channels to a value port, multiple business interfaces are mapped to value ports from a single value interface.

- An e^3*value* value interface is a collection of two or more ports that defines a commercial transaction. This is not modeled in an ArchiMate model, because transactionality of commercial transactions is not represented in ArchiMate.
- An e^3*value* value transfer can be mapped to an ArchiMate flow relation and to an ArchiMate serving relation with a direction that depends on the direction of the e^3*value* dependency path.

> An e^3*value* transfer is a willingness to transfer a value object from provider to requester [44]. A flow relationship is a transfer of information, goods or money between elements [114]. In addition, value transfers are part of a dependency path that starts from one or more needs. This determines which elements delivers a service to which other element (the serving relation) [114].

- A value transfer is associated with a *value object* in e^3*value.*

> A value object is an object that is of economic value for at least one other actor.
>
> We will not to represent the value object in ArchiMate. There is no way to accurately map a value object to a single concept in ArchiMate. A value object can be money, can be a physical object, the outcome of a service or even an experience [44]. The first concept to try to map the value object to is the value concept. It is defined as ‘something of relative worth, utility or importance

of a concept', but it is not a physical good or money. A value object can be mapped, in case of a physical good, to a product. A product is defined as a coherent collection of services and/or passive structure elements, accompanied by a contract/set of agreements, which is offered as a whole to (internal or external) customers [114]. In the explanation of the ArchiMate specification the authors also illustrate that physical goods can be represented with a product. Services are used to expose the product to the environment. In the case of money, the object money cannot be represented with a value concept or a product from ArchiMate, but the economic value of money can be represented with the value concept. These are two separate things.

Summarizing,when we combine all these arguments together, we opt not to map a value object from e^3*value* to ArchiMate. There is no way to accurately represent a value object in ArchiMate in all instances.

10.4 Case study

10.4.1 The company

Company X is responsible for building startups based on an acquisition of intellectual property. The main goal of organization is to increase the share value of the startup and finally sell the startup to other investors. Company X has three major value-adding activities: Scouting new technology, supporting startups during their growth, and selling matured startups. Support provided by X ranges from HRM, legal, financial administration, providing management, etc. We cannot provide exact details and the business model below differs somewhat from the actual business model. The company had an existing EA in place.

10.4.2 Application of the guidelines

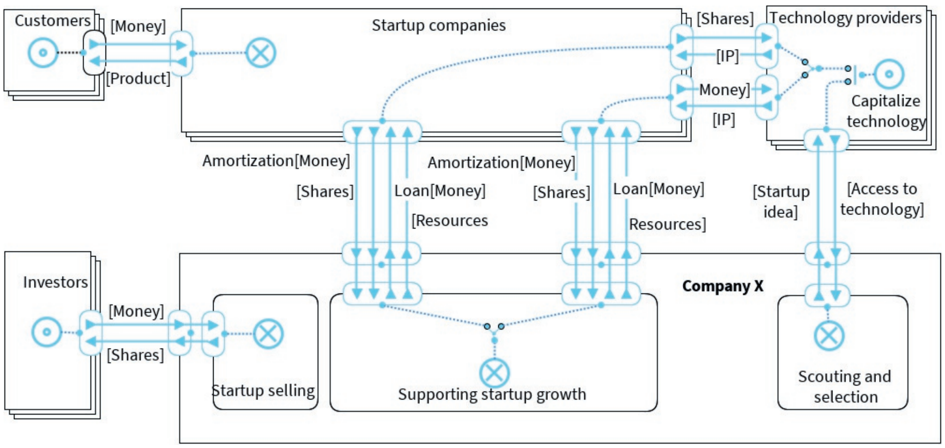


Figure 10.2: e^3*value* model of company X.

Figure 10.2 shows the business model of X. An e^3*value* model represents commercial transactions in a time period called the *contract period*.

In the model of figure 10.2, company X performs three value-adding activities, Scouting and selection, Supporting startup growth, and Startup selling. Technology providers have the need to capitalize technology. To satisfy this need, they exchange access to technology for a startup idea with company X and they transfer IP in the technology to a startup. IP can be transferred in exchange for shares or in exchange for money. In both cases, X lends money to the startup, makes resources available, receives startup shares and receives the amortization of the loan. However, only a fraction of the shares will be transferred if the technology provider receives part of the shares. This will be represented in the quantified model, discussed later.

Since this model contains no time ordering, it provides no information about when the loan is given, when the amortization takes place, or when the startup is sold. The model in figure 10.2 shows *all* of the commercial transactions and value activities that X is involved in during the contract period. These are the activities to be supported by an EA.

Figure 10.3 shows a high level layered EA model of X. The top part is constructed by our guidelines. We have also included a simple goal model. The diagrams have been annotated with the names of the guidelines applied.

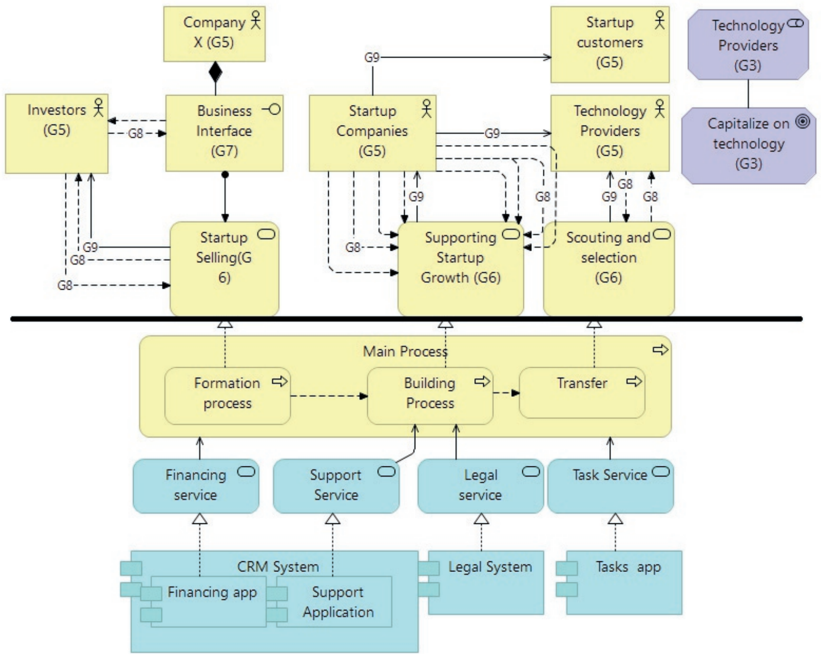


Figure 10.3: Layered EA model of X.

The part of the EA below the line had already been designed by the enterprise architect of X. Our guidelines produced the part of the enterprise architecture above the line. This part differs from the enterprise architect's model; most of it was absent. In particular, the business services were not complete and the external actors were absent.

For clarification reasons we will elaborate the ArchiMate figure. The top left part of the model is based on the left bottom part of the e^3*value* model. The startup selling value activity is mapped with rule G6 into a business service. The market segment investors is mapped with guideline G5 into a business actor. The flow relations are mapped from the value transfers using guidelines G8 and guideline G9 is used to identify a serves relation. We also mapped the value ports into a business interface using guideline G7. On the right side of the model, we mapped the technology provides with their needs into a small goal model using guideline G3.

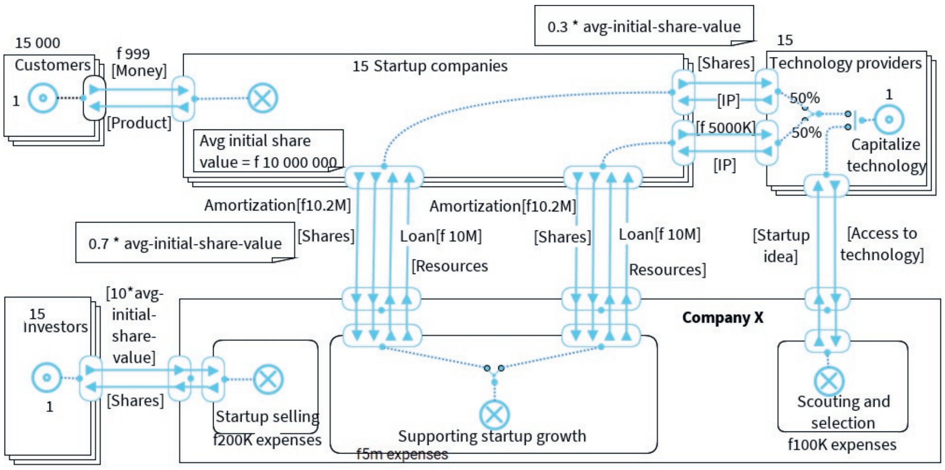


Figure 10.4: A market scenario for company X. "f"is a generic currency symbol.

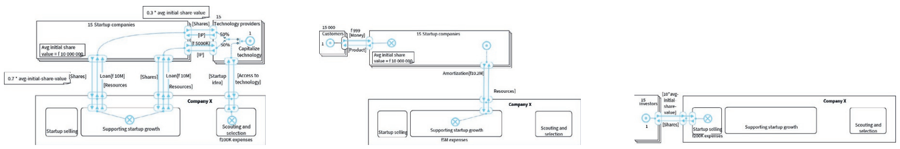
10.4.3 Quantification

A quantification of an e^3*value* model is called a *market scenario* [44]. For example, figure 10.4 shows a quantified version of the model in figure 10.2. It quantifies the size of market segments, the average number of needs per actor in a market segment, expenses of value activities, the value of money flows, the distribution of choices, and the value of any other variable that we introduced. Here, we introduce the initial share value of a startup as a variable. We use "f" as a generic currency symbol (read: "Florin"). The numbers in figure 10.4 are arbitrary and do not reflect company X.

Each actor has revenues and expenses, and adding these up, the e^3*value* tool computes that the company has a net revenue of f 747 M in this scenario. By making many different scenarios, we can assess how sensitive the business model is to differences in market assumptions.

In order to calculate the profitability of the focal company of the value network a time series can be used. A *time series* puts a number of market scenarios in chronological order and then calculates the profitability using a Net Prevent Value (NPV) calculation. Figure 10.5 shows what a time series looks like.[4] In the first period, X makes an investment in startups. In the

[4]Included to give a rough impression only. A version of the chapter with a readable time series is available at `https://www.thevalueengineers.nl/pdf/EMMSAD-2021-long.pdf`.



(a) Period 0: investment (b) Period 1: growth (c) Period 2: sale

Figure 10.5: Sketch of a time series for company X. A time series is a sequence of market scenarios for consecutive contract periods. Each market scenario quantifies an e3value model. In an investment analysis, the models in a time series usually are the same and the only thing that is different is the quantification in the consecutive market scenarios. However, for the computations that follow that is not important and we may create a time series where consecutive models are different. The three models are extracted from the market scenario in figure 10.4.

second period, these startups do business and in the third period, they are all sold to investors. Using the quantifications of figure 11.3 and a fictional interest rate of 2%, our tool computes an NPV in period 0 of f 571 M. By varying the quantifications, an investment risk analysis can be done.

10.4.4 Expert evaluations

In order to validate and elaborate on the correctness and utility of the realized traceability we organized a validation and requirements elicitation session with management and the enterprise architect of X.

To answer Q1, we made some mistakes in our initial business model, which we corrected. The mistakes were not in the value activities or the actors in the value network, but how actual value was created. For example, in our initial business model it is possible that not all shares are sold to X. In reality all shares are owned by X from startup creation. Also,the actual costs structure is completely different from our quantification. However, these mistakes have no impact on the resulting EA model.

The EA of figure 10.3 correctly represents the IT environment and business services of X and the upper part, designed by us, maps properly to the lower part, designed by the enterprise architect. This was also validated with the enterprise architect.

Our upper-part extension of the EA embodies an improved traceability from the EA to the e^3*value* business model, and the discussion revolved around answering Q2: Is this traceability useful for investment and divestment decisions?

This discussion turned into a requirements engineering session for tool support. Management of X wish to determine what the effect of changes in the business model on the EA is. Their strategic goal is to scale up to more start-ups than they have now (a quantitative goal) and they need to decide on the best investment in IT to support this goal. The value activities of X will not change but the number of technology providers and startups they interact with will change. Traceability between an e^3*value* model and an EA is a nice-to-have; the traceability will be considerably more useful to X if different quantified time series can be related to IT investments. X was particularly interested in being able to use NPV to evaluate different IT architectures from the business model in a top down manner and to be able to perform scenario generation and evaluation.

10.5 Discussion

10.5.1 Traceability

Our application of the guidelines showed that we can produce the upper part of an ArchiMate EA model that is a sound basis for designing a complete EA aligned to an e^3*value* model. Such an alignment is needed in order to relate IT expenses to value-adding business activities. All value-adding business services are included in the EA model, and they are related to interfaces with other companies. However, expert feedback told us that to be of use in investment decisions, this traceability relation should allow quantification.

10.5.2 A business model-driven method for EA design

Designing a business model and an EA requires many decisions and we found it expedient to use the following steps:

1. Construct an e^3*value* model for the value network of the focal organization.
2. Construct an ArchiMate motivation model.
    - Create a goal model for the organization using the guidelines of table 10.1.
    - Elaborate this goal model using ArchiMate guidelines and relations (composition, aggregation, specialization).

3. Construct an initial ArchiMate business layer model.
    - Construct a high level business service architecture (tables 10.2 and 10.3).
    - Identify sub-services where needed using standard ArchiMate modeling guidelines and operations (composition and aggregation).
4. Design the business processes and application architecture and link them to the service architecture.

We consider this as a lesson learned from this project and we will use this method for our next case study and to teach to students.

10.5.3 Validity

Internal validity is the extent to which the outcome of an experiment has been produced by the treatment. In this action experiment, the outcome is an EA and the treatment is our set of guidelines. Our description in this chapter shows that the outcome is indeed produced by this treatment.

The *utility* of this outcome is still an open question. The traceability that we established is a nice-to-have, but to be useful in investment and divestment decisions, we need to provide tool support to relate IT expenses to revenue in different investment scenarios.

Another open issue is the *external validity* of this treatment. Can other people use these guidelines and come up with similar results? Are these guidelines sufficient for all companies? Are the resulting EAs useful for other companies too? To answer these questions we need to do more case studies and experiments, in which we ask other people to use these guidelines for other companies.

A higher-level external validity question is whether guidelines like these can be used with other business modeling and EA languages. Achieving that level of generality is not our goal. Since our guidelines are derived from an analysis of the meta-models of e^3*value* and ArchiMate and refined in experiments and case studies using these languages, we do not expect generalizability beyond these languages.

10.6 Conclusion

We conclude that establishing traceability between an EA and an e^3*value* business model is possible in practice and is potentially useful if we can quantify this traceability relationship. In chapter 11, we will define a relationship between IT investments and company revenue and test this in a new case study. There is some previous research that we can build on [25, 83].

11

Quantitative Alignment of ArchiMate with e^3*value*[1]

[1]This chapter is based on an article in the proceedings of the International Conference on Conceptual Modeling[32]

11.1 Introduction

In chapters 8, 9 and 10 we defined and evaluated guidelines between e^3*value* and ArchiMate to create traceability between e^3*value* and ArchiMate. For this traceability to be useful we need to be able to assess the financial and technological sustainability of an enterprise architecture.

To assess financial sustainability of an ecosystem, we need a *business value model* of the ecosystem (henceforth called "business model"), which we define as a conceptual model that represents the creation, distribution, and capture of value in a *network* of participants [44]. Valuable objects are the outcome of services and products that satisfy customer needs, as well as payment for these.

A *quantified* business model of an ecosystem contains estimations of revenues and expenses of the ecosystem members. Revenues result from sales. Expenses are made to obtain e.g. raw materials, services or goods from others.

In ICT-intensive value propositions, expenses often relate to ICT components, both hard- and software. Therefore, in case of ICT-intensive services and products, the design of the provisioning *Enterprise Architecture* (EA) should be coordinated with business model design.

In this chapter we extend this with (1) guidelines to quantify workload requirements in an EA based on quantification of an e^3*value* model, (2) a technique by which to specify investments in and expenses on ICT in ArchiMate, and (3) a mechanism to import the specification of investments and expenses in e^3*value* models.

This chapter is structured as follows. Section 11.2 introduces our research methodology. In section 11.3 we introduce the design of our approach. In section 11.4 we apply our approach on a realistic example. We discuss our findings in section 11.5.

11.2 Design goals, research questions and methodology

Our design goal is to design techniques by which to determine if a business model is feasible in terms of financial sustainability and technological feasibility.

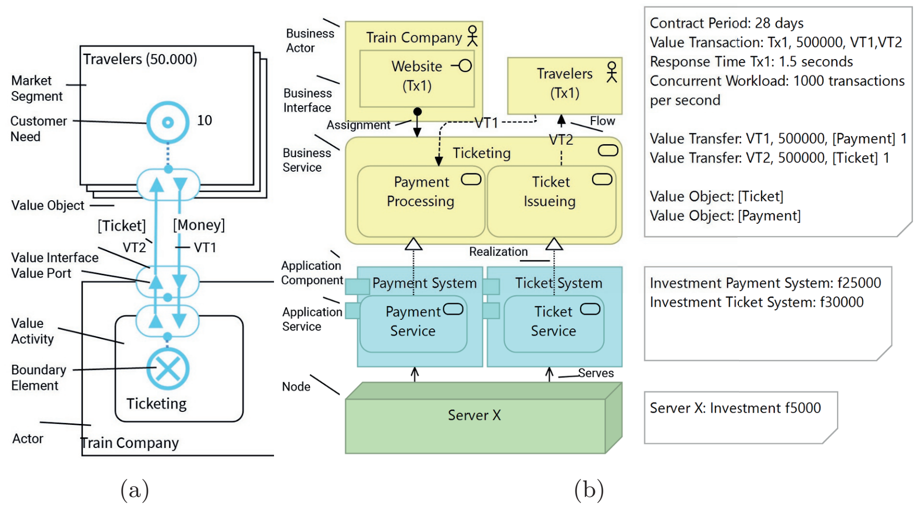
e^3*value* models contain a *transaction table* that identifies and counts all commercial transactions among ecosystem actors. The transaction table contains crucial information to assess long-term financial sustainability of the ecosystem because it determines revenues and expenses of each actor. Our first sub-goal is now to include information from the transaction table in an ArchiMate model. Our second sub-goal is to find a way to use this information to identify workload requirements on the components of an EA. Our third sub-goal is to find a way to specify investments and expenses on ICT in ArchiMate that will meet these workload requirements, and export these to the corresponding e^3*value* model. This gives us the following research questions.

- Q1: How can ArchiMate represent the economic transactions of e^3*value*?
- Q2: How can performance requirements in ArchiMate be identified from the transaction table?
- Q3: How can ArchiMate be quantified with investments and expenses?
- Q4: How can expenses and investments in an ArchiMate model be fed back into an e^3*value* model?
- Q5: Do these quantitative alignment techniques provide sufficient information for investment decisions?

Q1-Q4 are design questions, Q5 is a knowledge question. The aspect of usefulness that we want to consider in Q5 is *scalability.* In other words, are these techniques useful to make decisions about scaling up a given EA? We present our answers to Q1-Q4 in section 11.3 by means of a toy example and provide a preliminary answer to Q5 by means of a real-world case study in section 11.4. This means that we follow a design science methodology [127]. In our previous work we created guidelines for designing an ArchiMate business layer model from an e^3*value* model, based on a conceptual analysis of the two languages [27]. These guidelines where then tested and refined in a lab test and subsequently in a field test [28]. The current chapter is a further extension of the guidelines with quantification alignment between e^3*value* and ArchiMate and a preliminary field test of this extension.

11.3 Design of Quantitative Alignment

Figure11.1 contains the value network and transaction table of an e^3*value* model on the left and bottom, and an ArchiMate model on the right. We will explain all parts of the figure in what follows.



Actor	VA	Tx	#	VT	#	VO	VOo
Travelers	Ticketing	Tx1	500000	VT1	500000	Ticket	1
				VT2	500000	Money	1

Figure 11.1: e^3*value* model and ArchiMate model of a toy example.

Table 11.2: Guideline G9.

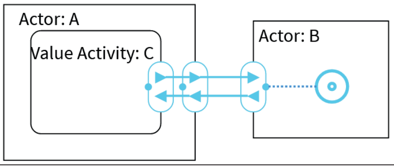
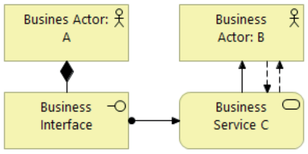
No	Guideline	Additional advice
G9	An e^3*value* activity connected through a value exchange to a need of an actor is mapped to an ArchiMate business service serving the actor. 	If B contains a boundary element instead of a need, the direction of the serving relation would be reversed.  Busines Actor: A Business Actor: B Business Interface Business Service C

Table 11.1: Definitions of e^3*value* and ArchiMate concepts. The first parts lists corresponding concepts. Using a business interface to represent a port is optional. Using a Serves relation to represent a value transfer is optional too.

e^3value	**Definition**	**ArchiMate**	**Definition**
Actor	An entity that is economically independent.	Business Actor	Business entity capable of performing behavior.
Value Activity	Profitable task performed by an actor.	Business Service	Defined behavior that is exposed to the environment.
Value Port	Willingness to provide or request value objects.	Business Interface	Channel that exposes behavior.
Value Transfer	Willingness to transfer value objects between actors.	Flow	Transfer from one element to another.
		Serves	Provide functionality to other element.
Other e^3*value* concepts			
Value Interface		Grouping of value ports .	
Value Object		An object that has economic value.	
Market Segment		A set of actors.	
Customer Need		Need to acquire something valuable.	
Boundary Element		Limit of value model.	
Other ArchiMate concepts			
Application component		Encapsulation of application functionality	
Flow		Transfer from one element to another .	
Assignment		Allocation of responsibility, performance of behavior.	
Application Service		Explicitly defined exposed application behavior.	
Realization		Realization of a more abstract entity.	
Node		A computational or physical resource.	

11.3.1 *e^3value*

Relevant definitions of *e^3value* and ArchiMate concepts are given in table 11.1. The *e^3value* model of figure 11.1 shows an *actor* train company exchanging *value objects* (tickets and money) with a *market segment* travelers. This is done through a *value activity* Ticketing.

To quantify an *e^3value* model, we use a so-called *contract period,* which is the period in which actors perform the transactions represented in the *e^3value* value network. A quantification says how large a market segment is, how often consumer needs occur, what the monetary value of money flows is, etc. In figure 11.1, there are 50 0000 travelers with each on the average 10 ticket needs in the contract period.

The *transaction table* at the bottom of figure 11.1 contains a quantification of the single transaction present in the value network. It says that transaction Tx1 occurs 500 000 times and consists of two value transfers, VT1 and VT2, through which tickets and money pass hands. These numbers are computed by the *e^3value* tool based on cardinality information provided by the tool user.

11.3.2 ArchiMate

The right part of figure 11.1 contains an ArchiMate model. The *business layer* of this model has been designed following the guidelines of our previous work. The crucial guideline G9 is shown in table 11.2. Figure 11.1 shows two ArchiMate actors, *Train Company* and *Travelers*, and a *Ticketing* service decomposed into two sub-services, *Payment Processing* and *Ticket Issuing.* In general, we define one (sub)service for each value transfer entering or leaving a business actor. In this EA, these services are implemented in two applications that run on the same server. We explain the remaining parts of figure 11.1 in the section that follows.

11.3.3 Representing the contract period in ArchiMate

To quantify an ArchiMate model, we need a contract period in ArchiMate too. Workloads, investments and expenses will refer to this contract period. We add the contract period to an ArchiMate model simply as a comment. Just as in *e^3value*, we can define a sequence of consecutive contract periods, called a *time series.* In figure 11.1 the duration of the contract period is 28 days.

11.3.4 Representing economic transactions in ArchiMate

In e^3*value* an economic transaction is created using *value ports*, *value interfaces* and *value transfers*. Except for value transfers, ArchiMate does not contain equivalent concepts. Therefore it is impossible to represent economic transactions in ArchiMate without extensions. To solve this, we add the information in an e^3*value* transaction table to ArchiMate models. In ArchiMate one can define attributes for model components. The collection of attributes defined for a component is called a *profile*.

- For each *value object*, we define an attribute of the ArchiMate model. The name of the attribute is the name of the value object. The EA in figure 11.1 has two value objects, *Ticket* and *Payment*.

- Each e^3*value* transfer corresponds to a flow in the ArchiMate model. For this flow we define a profile consisting of the attributes name, number of occurrences, and a reference to the value object. The two flows in the EA of figure 11.1 have value transfer profiles with attributes VT1, 500 000 occurrences, and value objects *Payment* and *Tickets*.

- Each transaction in e^3*value* consists of two or more value transfers, where each value transfer is part of a value interface of the two e^3*value* actors connected by the transfer. The connection points are *ports* in e^3*value*. In the corresponding EA, ports may be explicitly represented by *business interfaces* or implicitly by the incidence of a flow relation on an actor. This is a design choice of the ArchiMate model designer.

  We define a transaction profile for these business interfaces and actors, consisting of the transaction name, the number of occurrences, and references to the participating transfers. In figure 11.1. transaction Tx1 is defined for the business interface *Web Sit*e and for the business actor *Travelers*.

Figure 11.2 illustrates how value objects, value transfers and value transactions are mapped to the ArchiMate model.

11.3.5 Identification of additional performance requirements

In our approach we identify *performance requirements* derived from the transaction profiles assigned to the business layer elements of the ArchiMate model

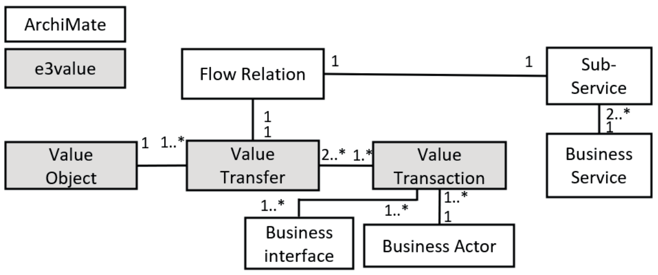


Figure 11.2: Relations among e^3value concepts (grey) and ArchiMate concepts (white).

and the *duration* of the contract period. The transaction profiles imply workload requirements.

The number of transactions in e^3value indicate the number of transactions that happen in a stated contract period, say one year. However, an e^3value model has no notion of time (except the contract period), so it does not model the distribution of transactions over the contract period. However, for technical scalability it is important to know this distribution, and more specifically the maximum number of transactions per second that can happen in the contract period. Therefore we define *peak economic transactions* requirements to represent this. These are additional to the requirements derived from the transaction table. Figure 11.1 contains a Concurrent Workload requirement as illustration.

In certain cases economic transactions needs to be completed within a time frame where the exchange of value objects is useful for the customer. Therefore, we also define *response time requirements*. This is based on work from IT performance metrics. Service performance is measured with the time it takes to execute a single instance of the service [116]. These are additional too. In figure 11.1 we see that Tx1 has a Response Time requirement.

The workload requirements are propagated from the business layer of ArchiMate over the actors that are needed to realize the business services. The response time requirements are propagated down to the business processes, application services and technology services.

11.3.6 Introducing Investments and Expenses in ArchiMate

e^3*value* has three types of quantifications for investment analysis [44]. First, *investments*. Investments are often needed when a new business idea is implemented. They are done in the first contract period of a time series and are subtracted from the revenue generate in that period.

Second, *fixed expenses*. These are the expenses that do not change from period to period, for example maintenance costs of IT systems. Fixed expenses can be specified for value activities, market segments and actors.

Finally *variable expenses*. These are the expenses associated with the execution of a single economic transaction, for example power consumption of IT systems, license fees, or acquisition of new hardware. The more economic transactions there are, the higher the total variable expenses are. Variable expenses are associated with value transfers, through the value ports in e^3*value*.

Our main strategy is to collect investments and expenses from an ArchiMate model and insert these into the e^3*value* model. ArchiMate [114] has so-called *internal active structure* elements, which are actors that can be hired, bought or built. For these, we define a profile consisting of the attributes *investment* and *fixed expenses*.

For *behavioral* elements (e.g. business processes and application services), we define the attribute *variable expenses*. The amount of these expenses depends on the number of executions of the behavior.

Finally, we define a profile consisting of the attributes *aggregated investments*, *aggregated fixed expenses*, and *aggregated variable expenses* for the ArchiMate business actors and services that correspond to e^3*value* actors and value activities. The investments in and expenses of the application and technology layers can be aggregated in these profiles and then transferred to the e^3*value* model. The aggregation can be done using the scripting languages of the ArchiMate modeling tools. Transfer to the e^3*value* model requires an update of the e^3*value* tool with an import function.

11.4 Case Study: Company X

11.4.1 The case company and its value model

Company X is responsible for building startups based on an acquisition of intellectual property, it has a portfolio of about 50 startups. The main goal of the organization is to increase the share value of the startup and finally sell the startup to other investors.

Figure 11.3 illustrates the e^3*value* model of company X. We will at the right side of the model. Company X (the focal company) has two value activities, automated spotting, and manual spotting. These value activities address a need at the technology providers. The market segment size is 3000, which means that there are 3000 possible technology providers. Company X main business model is to provide maturation services, like legal services, HRM services and even IT services (not in this model). Providing these kind of services increases the value of the startup companies they create. As soon as a startup company has reached a certain threshold it is sold to investors. Shares are exchanged for money. Startup companies also have a consumer need, they wish to acquire intellectual property and exchange this for money. Each startup company develops products based on this intellectual property and customers are willing to pay for this.

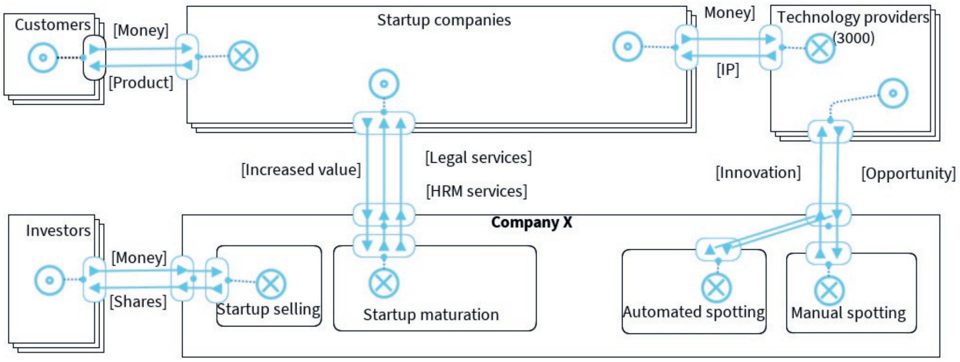
To validate the potential usefulness of our approach, we applied it to a company that is planning to upscale its business. We call the company X. Its business is to scout new technology, acquire the IP of that technology, create a startup for it, grow the startup and then sell it. Every year Company X identifies 3000 possible innovations based on pre-determined criteria per year. It uses a combination of automated technology spotting and manual technology spotting (figure 11.3).

For automated technology spotting they have developed bespoke technology. It accounts for 90 percent of all spotting activities. The split in spotting activities leads to two transactions. Automated spotting leads to transaction Tx1 and is executed 2700 times. Manual spotting leads to transaction Tx2, which is executed 300 times. These transactions are listed in figure 11.3.

11.4.2 Quantified Enterprise Architecture

Figure 11.4 illustrates the part of the EA for spotting innovations, based on the e^3*value* model from figure 11.3.The top annotations contain the results from importing the transaction table from e^3*value*. These now serve as workload requirements.

The architecture of Company X needs to support 2700x automated spotting and 300x manual spotting per 365 days. The transactions are broken up into the value transfers and via the sub-services they are propagated through the EA. For example, the internal search engine needs to find at least 2700 new innovations per 365 days and perform 2700 outreaches. The application service needs to support 2700 automated patent identifications and 300 manual patent identifications.



Actor	VA	Tx	#	VT	#	VO	VOo
Technology Providers	Automated Spotting	Tx1	2700	VT1	2700	Innovation	1
				VT2	2700	Opportunity	1
Technology Providers	Manual Spotting	Tx2	300	VT3	300	Innovation	1
				VT4	300	Opportunity	1

Figure 11.3: A market scenario for company X. "f"is a generic currency symbol. Pronounce "florin".

We have constructed the top part of the ArchiMate model with our guidelines, we refer to chapter 10 for the application of these guidelines. We have modeled Company X as a business actor, with two business interfaces (a nested composition). We have linked the business interfaces with the corresponding services, automated spotting, and manual spotting.

The market segment technology provides has also been translated into a business actor. Between the business services and the technology providers we see a flow relation from ArchiMate. We have annotated this flow relation with the corresponding value transfer from e^3*value*. To get a direct link between the value transfers and the rest of the enterprise architecture we need to decompose the main service into sub-services and link the value transfers with these sub-services. These sub-services can then be directly realized by other elements from the architecture. This way we realize traceability in such a way, that we can propagate the economic transactions as workload requirements over the architecture and collect investments and expenses in a bottom-up fashion to insert these into an e^3*value* model. A concrete implementation is out of scope for now, but we have annotated the model with illustrative investments and

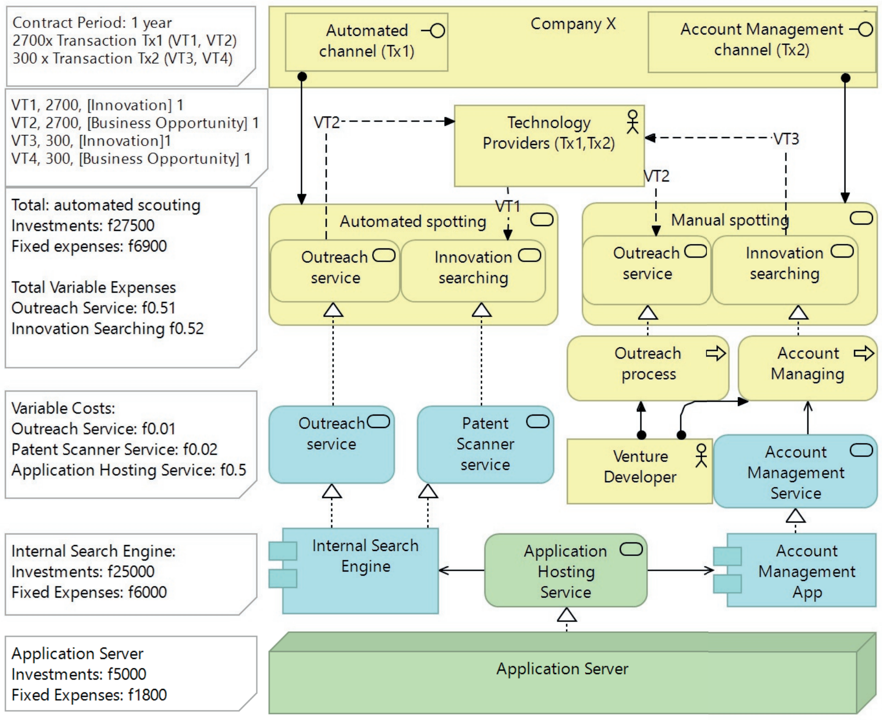


Figure 11.4: Architecture for innovation spotting

expenses for the automated spotting service. These are illustrated in the left part of the figure. The investments and fixed expenses are aggregated to the automated spotting service and from this inserted in the e^3*value* model through the automated spotting value activity. The variable expenses are aggregated to the services and this way linked to the value transfers associated with the flow relations. From this they are inserted into the e^3*value* model at the value transfer level.

11.4.3 Initial expert evaluation

We presented the quantified EA model to the enterprise architect of Company X and asked his opinion about the usefulness for their investment decisions. In his opinion, the quantitative alignment techniques are useful. He explained that Company X desires an approach where they can simulate the effects of different increases in economic transactions in the business model on the IT architecture. Using workload requirements based on economic transactions is a suitable way to reason about what the IT must support in that scenario according to the enterprise architect. It will help them identify the possible bottlenecks and provides a good overview of how far the economic transactions propagate down into the architecture. It will help them design the EA better.

The relation of investments and expenses with different scaling scenarios is particularly useful. It will help them get a breakdown of the investments and expenses for different scaling scenarios and improve decision making in investments decisions. The approach answers the questions where to invest and how expensive scaling would be related to the possible added benefits.

The enterprise architect also mentioned that he saw the most benefit in purely digital business models where the business model is completely realized in ICT. However, also in cases where people and business processes are used to implement a business model he saw an added benefit. The workload requirements could be propagated to the business process designers. Since Company X has a business model where parts are directly realized by IT and some by humans he was also interested to see the difference between the two variants in terms of expenses and scaling capabilities. Using operational data in the future is a nice to have. But using workload requirements is sufficient.

11.5 Discussion

11.5.1 Answers to research questions

ArchiMate can represent the economic transactions of e^3*value* (Q1) by exporting the transaction table into ArchiMate. The transactions are stored in custom profiles and assigned to ArchiMate elements. Performance requirements are identified (Q2) from the economic transactions and propagated down the ArchiMate model. Additionally peak concurrent transactions and response time requirements are identified as well. ArchiMate model elements can be extended (Q3) with a custom profile for investments and expenses and fed back into e^3*value*(Q4) because we have a custom profile aggregated expenses at the business services. These are mapped to the value activities and value transfers. Our initial evaluation suggested that these quantitative alignment techniques (Q5) provide sufficient information for investment decisions. It provided a breakdown of the investments and expenses needed per scaling scenario. It answers the question how expensive scaling is related to the possible additional benefits.

11.5.2 Validity

Internal validity is the extent to which the outcome of an experiment has been produced by the treatment. We followed the design guidelines that are our answers to Q1 - Q4 in our case study, so that at least we can say that this application of the guidelines produced the required outcome. The validation of the result by the enterprise architect (Q5) is encouraging but might also have been affected by the experimenter effect: Because the enterprise architect knew we had produced the models, he evaluated the models positively. We tried to reduce this threat to validity by asking specific questions about how the enterprise architect would use the results. Also, because we offered X to be a beta user of our tool implementation of these techniques, the enterprise architect has an interest in making this tool as useful as possible for them. This reduces the risk of an experimenter effect.

Another open issue is the *external validity* of this treatment. Can other people use these our method and come up with similar results? Does our method work for all companies? Are the resulting EAs useful for other companies too? To answer these questions we need to do more case studies and experiments, in which we ask other people to use these guidelines for other companies.

A higher-level external validity question is whether guidelines like these can be used with other business modeling and EA languages. Achieving that level of generality is not our goal. Since our results are derived from an analysis of the meta-models of e^3*value* and ArchiMate and refined in experiments and case studies using these languages, we do not expect generalizability beyond these languages.

11.5.3 Lessons learned

Realizing complete equivalence between ArchiMate and e^3*value* is not possible. Nor is it desirable. Combining all this information in one model would result in unmanageable models that are hard to understand. Separating commercial business models from EA models allows us to communicate with management and technical personnel separately. Keeping these models quantitatively aligned improves decision-making about ICT investments.

ArchiMate lacks constructs for generating economic transactions. However, using the profiling mechanism of ArchiMate the calculated transaction can be imported into ArchiMate and mapped to similar concepts. This way we do realize the required traceability between e^3*value* and ArchiMate.

Part IV

Conclusion

12

Conclusion

12.1 Evaluation of Design Goals

In chapter 1 we have stated our main design goals. The subsections of this chapter will discuss the realization in detail. The main goals of this research are:

1. To design and evaluate a traceability relation between enterprise architecture and the business goals
    - To be able to design and evaluate an enterprise architecture based on the goals of the stakeholders of the organization.
    - To be able to perform an impact of change analysis from the business goals to the enterprise architecture and back
    - To be able to perform a completeness analysis of the enterprise architecture based on business goal realization.
    - To evaluate the traceability relation in terms of understandability of the defined concepts.
2. To design a traceability relation between the business model and the enterprise architecture.
    - To evaluate the contribution of enterprise architecture to the business model.
    - To determine if a business model is feasible in terms of financial sustainability and technological feasibility.

12.1.1 Traceability between Enterprise Architecture and Business Goals

Our first design goal is that we wish to realize traceability between the enterprise architecture of an organization and its business goals. We have achieved this goal and its sub-goals. The first sub-goal is that we wish to have the ability to evaluate alternative EA designs based on the business goals. We have achieved this goal. By modeling alternative architectural solutions these alternatives can be illustrated and evaluated. Alternative solutions directions can be illustrated by having multiple realization relations. For example, we can have a design goal 'reduce number of support requests'. This goal can be satisfied in multiple ways, for example an online system, or better manuals. ARMOR introduces the ability in ArchiMate to link the problem domain with the solution domain.

The second sub-goal, that we wish to perform an impact of change analysis is also realized. We can both model the stakeholders, their goals and the realization of these goals by enterprise architecture elements. Therefore, a bi-directional impact of change analysis can be performed. We can select a stakeholder or business goal and use the traceability relations to identify all the elements in the enterprise architecture that are needed to satisfy this goal. A bottom-up analysis is also possible, where the traceability links are followed from the enterprise architecture into the goal models. This has also been implemented in the tool BiZZdesign Architect [35].

The third sub-goal is to have the ability to perform a completeness analysis of the enterprise architecture. If there are business goals not satisfied by the enterprise architecture, then either the enterprise architecture is incomplete, or the business goal is irrelevant. This helps the enterprise architect in determining how complete his designs are. We have constructed tool support for this in the tool BiZZdesign Architect in 2010 [35].

We have also evaluated the goal modeling part of ArchiMate in terms of understandability. We found that in order to increase understandability the number of concepts needs to be reduced. The concepts are conceptually very closely related and therefore introduce understandability issues when the language is used. We propose to simplify the language even further, and in case that is not possible, to introduce better guidelines for the practitioners.

12.1.2 Traceability between the Business Model and the Enterprise Architecture

Our second design goal is to realize traceability between the business model (modeled with e^3*value*) and the Enterprise Architecture (modeled with ArchiMate). We have achieved both sub-goals. The first goal, to evaluate the contribution of enterprise architecture to the business model, is realized through mapping business layer concepts to equivalent e^3*value* concepts. This introduces a qualitative traceability relation. This allows for reasoning why certain elements in the enterprise architecture are needed. This qualitative goal has been extended with a quantitative sub-goal, to determine if a business model is feasible in terms of financial sustainability and technological feasibility.

We import the economic transactions from an e^3*value* model into an ArchiMate model and propagate these as workload requirements. Based on these requirements an EA can be designed and evaluated. Second, investments and expenses can be identified and recorded in an ArchiMate model, collected and re-inserted into the e^3*value* model. e^3*value* can then use the financial quantifications to evaluate the feasibility of the proposed business model and resulting enterprise architecture.

12.2 Answering Main Research Questions

12.2.1 Research Questions

The individual chapters answered the research questions in detail. However, we will summarize the main research questions first and provide an answer to them.

- Q1: How can we extend ArchiMate with business goals to realize traceability?
- Q2: How well can practitioners use this extension? Which constructs do they use correctly, which incorrectly and why?
- Q3: How can we link an e^3*value* model with an ArchiMate model and how can we incorporate the quantifications of e^3*value* into ArchiMate?

12.2.2 Definition and initial validation of ARMOR

Research question Q1: how can we extend ArchiMate with business goals to realize traceability? is answered in chapter 3.

We have performed a conceptual analysis of existing literature like i*, KAOS and the business motivation model (BMM). We took essential concepts like stakeholder, hard goal, soft goal, requirement, goal decomposition, goal contribution and goal conflict from the literature. These were re-assembled into a new language, and we provided links with ArchiMate to realize the traceability. We also provide a first validation by applying ARMOR on an actual case study to demonstrate that we have achieved our design goals.

12.2.3 Understandability of ARMOR

Research Question Q2: how well can practitioners use this extension in practice? is answered in chapters 4,5 and 7. We organized multiple case studies in practice, where we validated the language with enterprise architects in chapter 4. Furthermore, we performed an understandability analysis of ARMOR, in chapters 5 and 7. We found out that only the concepts of stakeholder and goal were very well understood. Practitioners made modeling errors in the concepts of concern, and assessments. They also had a difficult time differentiating between goal and requirement. The full version of ARMOR was not very well understood. We concluded that this is because of the conceptual distance between the concepts. We provided a redesign of ARMOR with a reduced set of concepts.

12.2.4 Linking e^3*value* with ArchiMate

Research Question Q3: how can we link an e^3*value* model with an ArchiMate and how can we incorporate the quantifications of e^3*value* into ArchiMate. Chapters 8,9,10 and 11 answer this research question. We performed a conceptual analysis of both meta-models and structured an initial mapping. Value activities translate to business services, actors and market segments to business actors and stakeholders, value exchanges to flow relations and consumer needs to goals.

The economic transactions of e^3*value* are exported using the transaction table, we add attributes to the ArchiMate model and link the economic transactions to the business interfaces, flow relations and business services. We translate these economic transactions to workload requirements and then propagate these workload requirements over the enterprise architecture model. All elements capable of performing behavior have to realize this workload.

We also extend ArchiMate with attributes for investment, fixed expenses and variable expenses. Investments and fixed expenses are associated to the actors in the model. Variable expenses are associated with behavioral elements. This way variable expenses can be related to the value transfers and inserted into e^3*value*. Fixed expenses and variable expenses can be brought to the business services and inserted into e^3*value* at the value activity level.

12.3 Lessons Learned

Less can be more. During this execution of the first half of this thesis we evaluated a goal-oriented-requirements-engineering (GORE) extension of ArchiMate, called ARMOR. We evaluated this language through a few iterations in its development, but in essence it was a similar language with similar constructs. During this evaluation we found out that the language had understandability issues. We argued that the language contained many concepts that were semantically close, thus leading to understandability issues. One of the design goals of ARMOR is that it had to be a relatively simple language. During the execution of the validations the language was already simplified. The distinction between soft- and hard goal disappeared, and the conflict relation was removed. Still, this simplification was not sufficient to achieve a high level of understanding of the concepts. We therefore proposed an extreme simplification to ensure that the concepts were used correctly. Even during the initial design of ARMOR, we were already concerned with limiting the number of concepts, for example not modeling assumptions directly but as a property of a goal or finding a way to record the resolution of conflicts using existing constructs. This to limit the number of the concepts to increase concept understandability.

It is often easy to forget that an expressive language also requires a significant cognitive effort to correctly distinguish between all the elements. A formal definition of concepts would ensure that the concepts are at least well defined, but this still can lead to concepts that are semantically very close. For example, the difference between soft goal and hard goal or goal and requirement can be clearly defined, but can still lead to incorrect usage due to the fact that it takes cognitive effort to distinguish between closely related concepts.

We do need to remember that languages like ArchiMate, UML and the BPMN end up in hands of end-users who are often less capable, or not interested, in distinguishing al the subtle differences between the concepts. In the case of ARMOR and the motivation extension of ArchiMate 2.0 the difference between driver, assessment, goal and requirement was hard for practitioners to understand. The same conclusion can be drawn for i* [53], where concepts like soft-goal, hard-goal and task are also hard to understand for practitioners.

What is even more interesting to observe is that languages never seem to get less complex (and expressive), but always more complex. In the case of ArchiMate new concepts have been added to the motivation layer (e.g., outcome), and new layers have been added (e.g., strategy layer). The result is that it contains new additional concepts that are very closely related to already existing concepts.

During personal discussions with the author of i* and the current researchers during the i* workshop at *Conceptual Modeling 2019* the same could be observed. Instead of making the language simpler and more elegant, the main strategy was to make the language more expressive, without improving understandability. But the problem with these languages is that they are often already very expressive.

During the evaluations we did point out a few strategies that could be used to improve understandability. Our extreme simplification of ARMOR should be interpreted as a set of guidelines of a minimally usable sub-set of concepts that can be used to model most situations. If a limitation of concepts is not possible, we should provide guidelines instructing end-users which a sub-set of the language can be used in most instances similar to how BPMN level 1 and 2 are defined. The first level contains few concepts, but it is possible to model most processes of an organization. Understandability studies can also be used to improve the training material, they point out where the weak spots of the language are for the end-users. Training material can be adapted to better illustrate the differences between the concepts and teachers instructed to focus on those parts. The solution to this problem should lie somewhere between well defined concepts with sufficient distance between each other, restraint in designing the language regarding expressiveness, guidelines for usage, a more layered architecture like BPMN and good training material. But it should not be forgotten who the intended end-users are of the language. If the tool becomes too complex to use, the effectiveness of the tool can diminish.

12.3.1 Guidelines for business-model-driven EA design

In this thesis we have provided the basic building blocks to have the ability to evaluate alternative architectures implementing an e^3*value* model, but we have not provided guidelines to do so. Figure 12.1 illustrates the resulting guidelines for business-model driven EA design. The first step is to create an e^3*value* model of the value network. The second step is to quantify the e^3*value* model with market segment size, number of consumer needs and the expected valuations. Step three is to construct a transaction table, using the e^3*value* tool, and export this into an ArchiMate model. Multiple quantifications can be created to design multiple scenarios for evaluation. The fourth step is to identify/use the economic transactions as workload requirements for the enterprise architecture and identify (additional) stakeholders and elicit their goals and requirements (step 5). We recommend to use the light version of ARMOR for goal and requirements modeling. Afterwards the enterprise architecture needs to be designed based on these requirements, step 6. After the EA design quantify the EA model with investments and expenses (step 7) and export this model back into the e^3*value* editor (step 8). e^3*value* can then be used to evaluate the designed architecture in terms of financial feasibility.

12.4 Future Work

We have identified topics for future research based on the approach defined in this thesis. Summarizing, we need to validate and refine our approach in new iterations of the design cycle and develop tool support. Second, since our approach adds design decisions to a business model we would to investigate which design decisions belong to certain types of business models and finally,

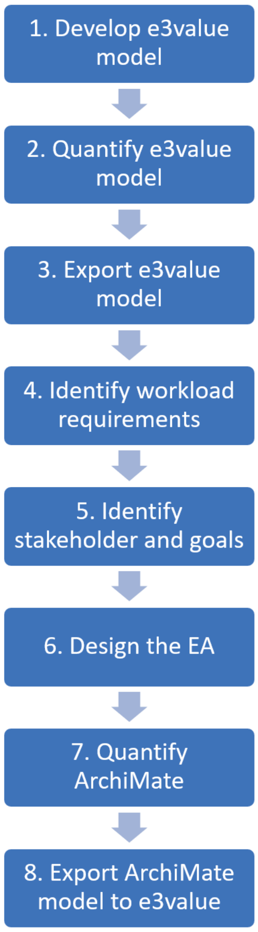

1. Develop e3value model
2. Quantify e3value model
3. Export e3value model
4. Identify workload requirements
5. Identify stakeholder and goals
6. Design the EA
7. Quantify ArchiMate
8. Export ArchiMate model to e3value

Figure 12.1: Guidelines for business-model driven enterprise architecture design. These guidelines provide a holistic link between different design phases. It is not meant as a concrete manual.

since we are in a transition to a circular economy, we need to investigate if our approach can be used in this context.

12.4.1 Validation and elaboration of the approach

Parts of our approach need more validation in practice. We would like to use a similar approach as our evaluation of the motivation extension of ArchiMate. We have created guidelines to align e^3*value* models with ArchiMate business layer and motivation layer diagrams. We wish to validate these guidelines in terms of understandability and utility. We want to know in more detail how useful our approach is for practitioners and if our guidelines are simple enough for them to use. We need to determine how well we can design an architecture based on workload requirements derived from the business model and if identifying investments and expenses in ArchiMate to elaborate and evaluate the business model is feasible. For example, we need to investigate how our approach works for IT systems directly realizing the economic transactions (i.e. when there is a direct relation between system usage and economic transactions) and for scenarios where there is an indirect relation between IT systems and the economic transactions.

A future step in this research is to integrate our approach in existing enterprise architecture design methods, like TOGAF [115].

Our proposed methodology is to keep using technical action research and start more iterations of the design cycle to further complete the validation.

12.4.2 Tool prototyping

Tool prototyping can be developed for our approach. Tool support for the integration of business goals in ArchiMate has already been developed. The first step is to adapt the e^3*value* web-editor[1]. We need to export an e^3*value* to an ArchiMate editor. This exported file should at least contain the transaction table of e^3*value* using the profiling mechanism of ArchiMate. Some of the concepts in e^3*value* may translate directly to ArchiMate equivalents, like business actors and business services.

Our recommendation is to use the free tool Archi[2]. Archi is an open source editor for ArchiMate models. It contains all the functionality we require. Archi supports the profiling language of ArchiMate. We will use this profiling language to import the e^3*value* transaction table and to propagate the economic transactions over the enterprise architecture as workload requirements.

Second, we need to use Archi to construct profiles for investments, fixed expenses and variable expenses. We can use the scripting language to develop

[1]`https://e3web.thevalueengineers.nl/login`

[2]`hhttps://www.archimatetool.com/`

algorithms for a bottom-up collection of these quantifications and automatically assign the aggregated results to the business services, business actors and value transfers in ArchiMate. The e^3*value* tool can then be extended/used to import this ArchiMate model and import the quantifications. These quantifications can be used to evaluate the financial feasibility of the enterprise architecture realizing the business model.

12.4.3 Operationalization of Business Models

In our approach a business model contains little to no design decisions regarding the operationalization of the business model, these design decisions are made in the design steps after business model design. In our approach we operationalize a business model with an enterprise architecture. We could investigate which kind of design decisions belong to a certain type of business model. This way a reference architecture could be identified based on the type of business model. This can supplement the field of reference architectures. In the field of enterprise architecture design a reference architecture is an accumulation of best practice architectures.

For example, for an insurance company there is a reference architecture containing the descriptions of how insurance companies look like. If we can refine this to types (or parts) of business models we can construct reference models for this. This would supplement the field of enterprise architecture and allow for fasting switching/adapting (parts) of the business model.

12.4.4 Circular Business Models

We view business model design from a value network perspective and this value network perspective will become more relevant during the transition to a circular economy. In a circular business model a value network contains a closing loop. After filling the loop with raw materials from traditional mining operations the majority of the raw material must be collected through recycling of old products. Organizations need to take back the old products, these products need to be transferred to a recycling plant, deconstructed into its raw materials, and then fed into the supply chain again. The goal is also to not only focus on creating value to the customers, while minimizing social and ecological costs [3].

This is supported by the literature [97]. Pieroni et al. reviewed the literature and found that business models should move away from the consumer

[3]https://sustainabilityguide.eu/methods/circular-business-models/

interface (i.e. only delivering value to the customer). The focus should be on the value network in a bi-directional manner with a holistic view. Business models should move away from the single organization view and abolish organizational boundaries. Business models should also contain quantifications other than costs and revenue structures. Finally, there should also be a shift from ownership to servitization. Consumers no longer own products but buy services around a certain tangible product. We believe that e^3*value* can play a role in designing circular business models.

We believe that our combination of a value network approach for business models and using enterprise architecture to design the organization around this value model can be used in this context. e^3*value* takes the networked holistic approach. This can be used to evaluate the economic sustainability of a circular business model for each actor in the model. e^3*value* can also be extended with other quantifications than just costs and revenues, like environmental impact. These quantifications must be propagated down or collected upwards from an enterprise architecture model similarly like we have done with the economic transactions, investments, expenses and variable expenses. For example, co2 emissions can be used, electricity usage, resource usage, etc in order to evaluate the feasibility of a business model.

We also believe that modeling the extended enterprise of a circular business model can assist in moving away from traditional organizational boundaries. The concept of an extended enterprise is that a company does not function in isolation, but in a value network. The extended enterprise can be used to focus on the collaboration required to deliver a product to a consumer, but also in a reverse direction. A circular business model contains more complex logistics than a linear business model and there are more interfaces between the different organizations. We also believe IT can assist in this.

In circular world, society will also impose design goals on organizations. These design goals are captured in the sustainability goals of the EU [4]. These goals can then be used to design a circular business model and the enterprise architecture. We wish to investigate what these effect of these design goals is on the design of the BM and the enterprise architecture.

12.5 Limitations

The limitations of this research are discussed in each individual chapter. We will summarize the main limitations in this section, and we will refer to the

[4]`https://ec.europa.eu/international-partnerships/sustainable-development-goals_en`

individual chapters.

The main limitation of our initial validation of ARMOR is that we performed single case studies. We generalized by analogy and we hypothesize that our results are also valid in similar organizations. This is discussed in section 4.4.4 and section 4.6.4. We also performed repeat studies, which confirmed our initial results. An interesting observation is that when the level of education and expertise in GORE notations increased, the same errors were made (but less often), see section 7.5.1. Based on these repeat studies, which included enterprise architects from different organizations, we are confident that our results regarding the understandability of ARMOR are valid.

This is supported by the research of Azevedo et al. [9]. They performed an ontological analysis and concluded that the concepts of the motivation extension of ArchiMate (which is a derivative of ARMOR) were conceptually too closely related. This confirms our results as well.

Because the goal-oriented concepts that we used have been taken from other existing goal-oriented languages, we hypothesize that our conclusions may be generalized to those languages too, section 5.7.1 discusses this in more detail.

The major limitation for realizing traceability between e^3*value* models and ArchiMate EA models is that we cannot claim generalizability to other languages, see section 8.5, section 9.7.1, and section 10.5.3. Generalizability to other languages was not our goal. Our guidelines have yet to be used by practitioners, as described in section 9.7.3.

The utility of traceability between e^3*value* and ArchiMate EA models also remains an open issue. We hypothesize that we can evaluate the technological and financial feasibility of a business model, but this needs more refinement and case studies, as discussed in section 10.5.3 and section 11.5.2.

Bibliography

[1] S. Abrahão, E. Insfran, J. A. Carsıé, M. Genero, and M. Piattini. “Evaluating the ability of novice analysts to understand requirements models”. In: *Quality Software, 2009. QSIC’09. 9th International Conference on*. IEEE. 2009, pp. 290–295.

[2] S. Abrahão, E. Insfran, J. A. Carsıé, and M. Genero. “Evaluating requirements modeling methods based on user perceptions: A family of experiments”. In: *Information Sciences* 181.16 (2011), pp. 3356–3378.

[3] A. Aldea, M. E. Iacob, J. van Hillegersberg, D. Quartel, and H. Franken. “Modelling value with archimate”. In: *Advanced Information Systems Engineering Workshops*. Springer. 2015, pp. 375–388.

[4] A. Aldea, M.-E. Iacob, and D. Quartel. “From Business Strategy to Enterprise Architecture and Back”. In: *2018 IEEE 22nd International Enterprise Distributed Object Computing Workshop (EDOCW)*. IEEE. 2018, pp. 145–152.

[5] R. Alt and H.-D. Zimmermann. “Preface: introduction to special section–business models”. In: *Electronic markets* 11.1 (2001), pp. 3–9.

[6] B. Andersson, M. Bergholtz, A. Edirisuriya, T. Ilayperuma, P. Jayaweera, P. Johannesson, and J. Zdravkovic. “Enterprise sustainability through the alignment of goal models and business models”. In: *Proceedings of 3rd International Workshop on Business/IT-Alignment and Interoperability (BUSITAL’08) CEUR Workshop Proceedings*. 2008.

[7] A. I. Anton. “Goal-based requirements analysis”. In: *Requirements Engineering, 1996., Proceedings of the Second International Conference on*. IEEE. 1996, pp. 136–144.

[8] C. L. Azevedo, J. P. A. Almeida, M. van Sinderen, D. Quartel, and G. Guizzardi. “An ontology-based semantics for the motivation extension to archimate”. In: *2011 IEEE 15th International Enterprise Distributed Object Computing Conference*. IEEE. 2011, pp. 25–34.

[9] C. L. Azevedo, J. P. A. Almeida, M. van Sinderen, D. Quartel, and G. Guizzardi. “An ontology-based semantics for the motivation extension to archimate”. In: *2011 IEEE 15th International Enterprise Distributed Object Computing Conference*. IEEE. 2011, pp. 25–34.

[10] V. A. Batista, D. C. Peixoto, W. Pádua, and C. I. P. Pádua. “Using UML stereotypes to support the requirement engineering: a case study”. In: *Computational Science and Its Applications–ICCSA 2012*. Springer, 2012, pp. 51–66.

[11] P. v. Bommel, P. Buitenhuis, S. Hoppenbrouwers, and E. Proper. “Architecture principles–A regulative perspective on enterprise architecture”. In: *Enterprise modelling and information systems architectures–concepts and applications* (2007).

[12] C. Braun and R. Winter. “A comprehensive enterprise architecture metamodel and its implementation using a metamodeling platform”. In: (2005).

[13] P. Bresciani, A. Perini, P. Giorgini, F. Giunchiglia, and J. Mylopoulos. “Tropos: An agent-oriented software development methodology”. In: *Autonomous Agents and Multi-Agent Systems* 8.3 (2004), pp. 203–236.

[14] C. Britton, M. Kutar, S. Anthony, and T. Barker. “An empirical study of user preference and performance with UML diagrams”. In: *Human Centric Computing Languages and Environments, 2002. Proceedings. IEEE 2002 Symposia on*. IEEE. 2002, pp. 31–33.

[15] F. P. Brooks and N. S. Bullet. “Essence and accidents of software engineering”. In: *IEEE computer* 20.4 (1987), pp. 10–19.

[16] Business Motivation Model. “Business Motivation Model Version 1.0”. In: *Standard document URL: http://www. omg. org/spec/BMM/1.0/PDF (22.09. 2009)* (2007).

[17] R. van Buuren, J. Gordijn, and W. Janssen. “Business case modelling for E-services”. In: *BLED 2005 Proceedings* (2005), p. 8.

[18] A. Caetano, G. Antunes, J. Pombinho, M. Bakhshandeh, J. Granjo, J. Borbinha, and M. M. Da Silva. “Representation and analysis of enterprise models with semantic techniques: an application to ArchiMate, e3value and business model canvas”. In: *Knowledge and Information Systems* 50.1 (2017), pp. 315–346.

[19] P. Caire, N. Genon, D. Moody, et al. “Visual Notation Design 2.0: Towards User-Comprehensible RE Notations”. In: *Proceedings of the 21st IEEE International Requirements Engineering Conference*. 2013.

[20] J. P. Carvallo and X. Franch. “On the use of i* for Architecting Hybrid Systems: A Method and an Evaluation Report”. In: *The Practice of Enterprise Modeling*. Springer, 2009, pp. 38–53.

[21] P. Clements and L. Bass. “Using Business Goals to Inform a Software Architecture”. In: *18th IEEE International Requirements Engineering Conference*. IEEE Computer Society Press. 2010, pp. 69–78.

[22] J. A. Cruz-Lemus, M. Genero, M. E. Manso, S. Morasca, and M. Piattini. “Assessing the understandability of UML statechart diagrams with composite states: A family of empirical studies”. In: *Empirical Software Engineering* 14.6 (2009), pp. 685–719.

[23] A. Dardenne, A. v. Lamsweerde, and S. Fickas. “Goal-directed requirements acquisition”. In: *Science of Computer Programming* 20.1–2 (1993), pp. 3–50.

[24] A. De Lucia, C. Gravino, R. Oliveto, and G. Tortora. “An experimental comparison of ER and UML class diagrams for data modelling”. In: *Empirical Software Engineering* 15.5 (2010), pp. 455–492.

[25] Z. Derzsi, J. Gordijn, K. Kok, H. Akkermans, and Y.-H. Tan. “Assessing feasibility of IT-enabled networked value constellations: A case study in the electricity sector”. In: *International Conference on Advanced Information Systems Engineering*. Springer. 2007, pp. 66–80.

[26] J. L. Dietz. “Understanding and modelling business processes with DEMO”. In: *International Conference on Conceptual Modeling*. Springer. 1999, pp. 188–202.

[27] W. Engelsman, R. J. Wieringa, M. van Sinderen, J. Gordijn, and T. Haaker. “Realizing Traceability from the Business Model to Enterprise Architecture”. In: *Advances in Conceptual Modeling*. Springer. 2019, pp. 37–46.

[28] W. Engelsman, R. J. Wieringa, M. van Sinderen, J. Gordijn, and T. Haaker. “Transforming e3value models in ArchiMate diagrams”. In: *2020 IEEE 24th International Enterprise Distributed Object Computing Conference*. Springer. 2020, pp. 11–20.

[29] W. Engelsman, D. A. C. Quartel, H. Jonkers, and M. J. van Sinderen. “Extending enterprise architecture modelling with business goals and requirements”. In: *Enterprise information systems* 5.1 (2011), pp. 9–36. ISSN: 1751-7575.

[30] W. Engelsman and R. J. Wieringa. “Understandability of goal-oriented requirements engineering concepts for enterprise architects”. In: *Advanced Information Systems Engineering (CAiSE), 26th international conference*. Springer, 2014.

[31] W. Engelsman, J. Gordijn, T. Haaker, M. van Sinderen, and R. Wieringa. “Traceability from the Business Value Model to the Enterprise Architecture: A Case Study”. In: *Enterprise, Business-Process and Information Systems Modeling*. Springer, 2021, pp. 212–227.

[32] W. Engelsman, J. Gordijn, T. Haaker, M. v. Sinderen, and R. Wieringa. “Quantitative Alignment of Enterprise Architectures with the Business Model”. In: *International Conference on Conceptual Modeling*. Springer. 2021, pp. 189–198.

[33] W. Engelsman, M. E. Iacob, and H. M. Franken. “Architecture-driven requirements engineering”. In: *Proceedings of the 2009 ACM symposium on Applied Computing*. 2009, pp. 285–286.

[34] W. Engelsman, H. Jonkers, H. M. Franken, and M.-E. Iacob. “Architecture-driven requirements engineering”. In: *Working Conference on Practice-Driven Research on Enterprise Transformation*. Springer. 2009, pp. 134–154.

[35] W. Engelsman, H. Jonkers, and D. Quartel. “ArchiMate® extension for modeling and managing motivation, principles, and requirements in TOGAF®”. In: *White paper, The Open Group* (2011).

[36] W. Engelsman and R. Wieringa. “Goal-Oriented requirements engineering and enterprise architecture: two case studies and some lessons learned”. In: *Requirements Engineering: Foundation for Software Quality*. Springer, 2012, pp. 306–320.

[37] W. Engelsman and R. Wieringa. “Understandability of goal concepts by requirements engineering experts”. In: *Advances in Conceptual Modeling*. Springer. 2014, pp. 97–106.

[38] A. Finkelstein, J. Kramer, and M. Goedicke. *Viewpoint oriented software development*. University of London, Imperial College of Science and Technology, Department . . ., 1991.

[39] J. Forrester. “If p, then what? Thinking in cases”. In: *History of the human sciences* 9.3 (1996), pp. 1–25.

[40] B. Fritscher and Y. Pigneur. “Business IT alignment between business model and enterprise architecture with a strategic perspective”. In: *International Journal of Information System Modeling and Design* 6.1 (2015), pp. 1–23.

[41] B. Fritscher and Y. Pigneur. “Business IT alignment from business model to enterprise architecture”. In: *International Conference on Advanced Information Systems Engineering*. Springer. 2011, pp. 4–15.

[42] I. Galvao and A. Goknil. “Survey of traceability approaches in model-driven engineering”. In: *11th IEEE International Enterprise Distributed Object Computing Conference (EDOC 2007)*. IEEE. 2007, pp. 313–313.

[43] S. Ghaisas, P. Rose, M. Daneva, and K. Sikkel. “Generalizing by similarity: Lessons learnt from industrial case studies”. In: *1st International Workshop on Conducting Empirical Studies in Industry (CESI)*. IEEE Computer Science Press, 2013, pp. 37–42.

[44] J. Gordijn and R. Wieringa. e^{3}value*User guide*. `https://e3value-user-manual.thevalueengineers.nl/`. The Value Engineers, 2021.

[45] J. Gordijn. “E-business value modelling using the e3-value ontology”. In: *Value creation from e-business models*. Elsevier, 2004, pp. 98–127.

[46] J. Gordijn and J. Akkermans. “Value-based requirements engineering: exploring innovative e-commerce ideas”. In: *Requirements engineering* 8.2 (2003), pp. 114–134.

[47] J. Gordijn, P. Van Eck, and R. Wieringa. “Requirements engineering techniques for e-services”. In: *Service-Oriented Computing: Cooperative Information Systems Series* (2009), pp. 331–352.

[48] J. Gordijn, E. Yu, and B. Van Der Raadt. “E-service design using i* and e/sup 3/value modeling”. In: *IEEE software* 23.3 (2006), pp. 26–33.

[49] J. Gordijn and J. M. Akkermans. *Value Webs: Understanding E-business Innovation*. `www.thevalueengineers.nl`. The Value Engineers, 2018.

[50] I. Hadar, I. Reinhartz-Berger, T. Kuflik, A. Perini, F. Ricca, and A. Susi. “Comparing the Comprehensibility of Requirement Models Expressed in Use Case and Tropos: Results from a Family of Experiments”. In: *Information and Software Technology* (2013).

[51] C. L. Heitmeyer, R. D. Jeffords, and B. G. Labaw. "Automated consistency checking of requirements specifications". In: *ACM Transactions on Software Engineering and Methodology (TOSEM)* 5.3 (1996), pp. 231–261.

[52] J. Horkoff and E. Yu. "Evaluating Goal Achievement in Enterprise Modeling–An Interactive Procedure and Experiences". In: *The Practice of Enterprise Modeling*. Springer, 2009, pp. 145–160.

[53] C. Houy, P. Fettke, and P. Loos. *Understanding understandability of conceptual models - what are we actually talking about? - Supplement*. eng. Tech. rep. Postfach 151141, 66041 Saarbrucken: Universitat und Landesbibliothek, 2013. URL: `http://scidok.sulb.uni-saarland.de/volltexte/2013/5441`.

[54] M.-E. Iacob and H. Jonkers. "Quantitative analysis of enterprise architectures". In: *Interoperability of Enterprise Software and Applications*. Springer, 2006, pp. 239–252.

[55] M.-E. Iacob, L. O. Meertens, H. Jonkers, D. A. Quartel, L. J. Nieuwenhuis, and M. J. Van Sinderen. "From enterprise architecture to business models and back". In: *Software & Systems Modeling* 13.3 (2014), pp. 1059–1083.

[56] M.-E. Iacob, D. Quartel, and H. Jonkers. "Capturing business strategy and value in enterprise architecture to support portfolio valuation". In: *2012 IEEE 16th International Enterprise Distributed Object Computing Conference*. IEEE. 2012, pp. 11–20.

[57] "ISO/IEC/IEEE Systems and software engineering – Architecture description". In: *ISO/IEC/IEEE 42010:2011(E) (Revision of ISO/IEC 42010:2007 and IEEE Std 1471-2000)* (2011), pp. 1–46. DOI: `10.1109/IEEESTD.2011.6129467`.

[58] I. Jacobson, G. Booch, and J. Rumbaugh. "The Unified Modeling Language". In: *University Video Communications* (1996).

[59] H. Jonkers, M. Lankhorst, R. Van Buuren, S. Hoppenbrouwers, M. Bonsangue, and L. Van Der Torre. "Concepts for modeling enterprise architectures". In: *International Journal of Cooperative Information Systems* 13.3 (2004), pp. 257–287. ISSN: 0218-8430.

[60] I. Jureta and S. Faulkner. "An agent-oriented meta-model for enterprise modelling". In: *Perspectives in Conceptual Modeling*. Springer, 2005, pp. 151–161.

[61] E. Kamsties, A. von Knethen, and R. Reussner. “A controlled experiment to evaluate how styles affect the understandability of requirements specifications”. In: *Information and Software Technology* 45.14 (2003), pp. 955–965.

[62] F. Kaya, J. Gordijn, R. Wieringa, and M. Makkes. “Governance in peer-to-peer networks is a design problem”. In: *14th International Workshop on Value Modelling and Business Ontologies, VMBO 2020*. CEUR-WS. 2020, pp. 125–132.

[63] W. C. Kim and R. A. Mauborgne. *Blue ocean strategy, expanded edition: How to create uncontested market space and make the competition irrelevant*. Harvard business review Press, 2014.

[64] S. de Kinderen, K. Gaaloul, and H. E. Proper. “Integrating value modelling into ArchiMate”. In: *International Conference on Exploring Services Science*. Springer. 2012, pp. 125–139.

[65] S. de Kinderen, K. Gaaloul, and H. A. Proper. “Bridging value modelling to ArchiMate via transaction modelling”. In: *Software & Systems Modeling* 13.3 (2014), pp. 1043–1057.

[66] B. Kitchenham. “Procedures for performing systematic reviews”. In: *Keele, UK, Keele University* 33 (2004), p. 2004.

[67] S. Kurpjuweit and R. Winter. “Concern-oriented business architecture engineering”. In: *Proceedings of the 2009 ACM symposium on Applied Computing*. 2009, pp. 265–272.

[68] M. Kutar, C. Britton, and T. Barker. “A comparison of empirical study and cognitive dimensions analysis in the evaluation of UML diagrams”. In: *Procs of the 14th Workshop of the Psychology of Programming Interest Group (PPIG 14)*. 2002.

[69] L. Kuzniarz, M. Staron, and C. Wohlin. “An empirical study on using stereotypes to improve understanding of UML models”. In: *Program Comprehension, 2004. Proceedings. 12th IEEE International Workshop on*. IEEE. 2004, pp. 14–23.

[70] R. Lagerström, J. Saat, U. Franke, S. Aier, and M. Ekstedt. “Enterprise meta modeling methods–combining a stakeholder-oriented and a causality-based approach”. In: *Enterprise, business-process and information systems modeling*. Springer, 2009, pp. 381–393.

[71] C. F. Lange and M. R. Chaudron. “Interactive views to improve the comprehension of UML models-an experimental validation”. In: *Program Comprehension, 2007. ICPC’07. 15th IEEE International Conference on.* IEEE. 2007, pp. 221–230.

[72] M. Lankhorst. *Enterprise architecture at work: Modelling, communication and analysis.* Springer-Verlag New York Inc, 2009. ISBN: 3642013090.

[73] M. M. Lankhorst, H. A. Proper, and H. Jonkers. “The architecture of the archimate language”. In: *Enterprise, business-process and information systems modeling.* Springer, 2009, pp. 367–380.

[74] S. Lauesen. *User interface design: a software engineering perspective.* Pearson Education, 2005.

[75] D. Leslie and N. M. Rantisi. “Creativity and place in the evolution of a cultural industry: the case of Cirque du Soleil”. In: *Urban Studies* 48.9 (2011), pp. 1771–1787.

[76] A. Lindstrom. “On the syntax and semantics of architectural principles”. In: *Proceedings of the 39th Annual Hawaii International Conference on System Sciences (HICSS’06).* Vol. 8. IEEE. 2006, 178b–178b.

[77] I. Malavolta, P. Lago, H. Muccini, P. Pelliccione, and A. Tang. “What industry needs from architectural languages: A survey”. In: *IEEE Transactions on Software Engineering* 39.6 (2012), pp. 869–891.

[78] A. Maté, J. Trujillo, and X. Franch. “A modularization proposal for goal-oriented analysis of data warehouses using I-star”. In: *Conceptual Modeling–ER 2011.* Springer, 2011, pp. 421–428.

[79] R. Matulevičius and P. Heymans. “Comparing goal modelling languages: An experiment”. In: *Requirements Engineering: Foundation for Software Quality.* Springer, 2007, pp. 18–32.

[80] V. Mayer-Schönberger and T. Ramge. *Reinventing capitalism in the age of big data.* Basic Books, 2018.

[81] L. O. Meertens, M.-E. Iacob, L. J. Nieuwenhuis, M. J. van Sinderen, H. Jonkers, and D. Quartel. “Mapping the business model canvas to ArchiMate”. In: *Proceedings of the 27th annual ACM symposium on applied computing.* ACM. 2012, pp. 1694–1701.

[82] T. Mens and P. Van Gorp. “A taxonomy of model transformation”. In: *Electronic notes in theoretical computer science* 152 (2006), pp. 125–142.

[83] J. Miguens, M. M. da Silva, and S. Guerreiro. “A Viewpoint for Integrating Costs in Enterprise Architecture”. In: *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*. Springer. 2018, pp. 481–497.

[84] G. A. Miller. “The magical number seven, plus or minus two: some limits on our capacity for processing information.” In: *Psychological review* 63.2 (1956), p. 81.

[85] D. Moody. “The Physics of Notations: Improving the Usability and Communicability of Visual Notations in Requirements Engineering”. In: *Requirements Engineering Visualization (REV), 2009 Fourth International Workshop on.* 2009, pp. 56–57. DOI: 10.1109/REV.2009.6.

[86] D. Moody, P. Heymans, and R. Matulevicius. “Improving the Effectiveness of Visual Representations in Requirements Engineering: An Evaluation of i* Visual Syntax”. In: *Requirements Engineering Conference, 2009. RE'09. 17th IEEE International.* IEEE Computer Society Press, 2009, pp. 171–180.

[87] D. Moody. “The "Physics" of Notations: Toward a Scientific Basis for Constructing Visual Notations in Software Engineering”. In: *Software Engineering, IEEE Transactions on* 35.6 (2009), pp. 756–779. ISSN: 0098-5589. DOI: 10.1109/tse.2009.67.

[88] D. L. Moody, P. Heymans, and R. Matulevičius. “Visual syntax does matter: improving the cognitive effectiveness of the i* visual notation”. In: *Requirements Engineering* 15.2 (2010), pp. 141–175.

[89] J. F. Moore. *The Death of Competition: Leadership and Strategy in the Age of Business Ecosystems.* Harper, 1996.

[90] M. Morandini, A. Marchetto, and A. Perini. “Requirements comprehension: A controlled experiment on conceptual modeling methods”. In: *Empirical Requirements Engineering (EmpiRE), 2011 First International Workshop on.* IEEE. 2011, pp. 53–60.

[91] A. Nugroho. “Level of detail in UML models and its impact on model comprehension: A controlled experiment”. In: *Information and Software Technology* 51.12 (2009), pp. 1670–1685.

[92] M. Op't Land and H. Proper. “Impact of principles on enterprise engineering”. In: (2007).

[93] A. Osterwalder and Y. Pigneur. *Business model generation: a handbook for visionaries, game changers, and challengers.* John Wiley & Sons, 2010.

[94] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee. “A design science research methodology for information systems research”. In: *Journal of management information systems* 24.3 (2007), pp. 45–77.

[95] R. M. Pessoa, M. van Sinderen, and D. A. Quartel. “Towards Requirements Elicitation in Service-oriented Business Networks using Value and Goal Modelling.” In: *ICSOFT (2)*. 2009, pp. 392–399.

[96] J. Petrikina, P. Drews, I. Schirmer, and K. Zimmermann. “Integrating business models and enterprise architecture”. In: *2014 IEEE 18th International Enterprise Distributed Object Computing Conference Workshops and Demonstrations*. IEEE. 2014, pp. 47–56.

[97] M. P. Pieroni, T. C. McAloone, and D. C. Pigosso. “Business model innovation for circular economy and sustainability: A review of approaches”. In: *Journal of cleaner production* 215 (2019), pp. 198–216.

[98] M. E. Porter. “Competitive advantage, agglomeration economies, and regional policy”. In: *International regional science review* 19.1-2 (1996), pp. 85–90.

[99] H. C. Purchase, L. Colpoys, M. McGill, D. Carrington, and C. Britton. “UML class diagram syntax: an empirical study of comprehension”. In: *Proceedings of the 2001 Asia-Pacific symposium on Information visualisation-Volume 9*. Australian Computer Society, Inc. 2001, pp. 113–120.

[100] H. C. Purchase, R. Welland, M. McGill, and L. Colpoys. “Comprehension of diagram syntax: an empirical study of entity relationship notations”. In: *International Journal of Human-Computer Studies* 61.2 (2004), pp. 187–203. ISSN: 1071-5819. DOI: `http://dx.doi.org/10.1016/j.ijhcs.2004.01.003`. URL: `http://www.sciencedirect.com/science/article/pii/S1071581904000072`.

[101] D. A. C. Quartel, W. Engelsman, H. Jonkers, and M. J. van Sinderen. “A goal-oriented requirements modelling language for enterprise architecture”. In: *Proceedings of the Thirteenth IEEE International EDOC Enterprise Computing Conference, EDOC 2009, Auckland, New Zealand*. Auckland, New Zealand: IEEE Computer Society Press, 2009, pp. 3–13.

[102] B. Ramesh and M. Jarke. “Toward reference models for requirements traceability”. In: *IEEE transactions on software engineering* 27.1 (2001), pp. 58–93.

[103] G. Reggio, F. Ricca, G. Scanniello, F. Di Cerbo, and G. Dodero. “A precise style for business process modelling: Results from two controlled experiments”. In: *Model Driven Engineering Languages and Systems.* Springer, 2011, pp. 138–152.

[104] D. T. Ross and K. E. Schoman. “Structured analysis for requirements definition”. In: *IEEE transactions on Software Engineering* 1 (1977), pp. 6–15.

[105] K. Siau and P.-P. Loo. “Identifying difficulties in learning UML”. In: *Information Systems Management* 23.3 (2006), pp. 43–51.

[106] R. G. Slot. “A method for valuing architecture-based business transformation and measuring the value of solutions architecture”. PhD thesis. Universiteit van Amsterdam [Host], 2010.

[107] Z. Soh, Z. Sharafi, B. Van den Plas, G. C. Porras, Y.-G. Guéhéneuc, and G. Antoniol. “Professional status and expertise for UML class diagram comprehension: An empirical study”. In: *Program Comprehension (ICPC), 2012 IEEE 20th International Conference on.* IEEE. 2012, pp. 163–172.

[108] D. Stelzer. “Enterprise architecture principles: literature review and research directions”. In: *Service-oriented computing. ICSOC/ServiceWave 2009 workshops.* Springer. 2009, pp. 12–21.

[109] J. Stirna, A. Persson, and K. Sandkuhl. “Participative enterprise modeling: experiences and recommendations”. In: *Advanced Information Systems Engineering.* Springer. 2007, pp. 546–560.

[110] H. Storrle. “On the impact of layout quality to understanding UML diagrams”. In: *Visual Languages and Human-Centric Computing (VL/HCC), 2011 IEEE Symposium on.* IEEE. 2011, pp. 135–142.

[111] C. R. Sunstein. “On analogical reasoning”. In: *Harvard Law Review* 106.3 (1993), pp. 741–791.

[112] D. J. Teece. “Business models, business strategy and innovation”. In: *Long range planning* 43.2-3 (2010), pp. 172–194.

[113] M. A. Teruel, E. Navarro, V. López-Jaquero, F. Montero, J. Jaen, and P. González. “Analyzing the understandability of Requirements Engineering languages for CSCW systems: A family of experiments”. In: *Information and Software Technology* 54.11 (2012), pp. 1215–1228.

[114] The Open Group. *ArchiMate 3.1 Specification.* Van Haren Publishing, 2019.

[115] The Open Group. *TOGAF Version 9*. Van Haren Publishing, 2009.

[116] I. Traore, I. Woungang, A. A. E. S. Ahmed, and M. S. Obaidat. "Software Performance Modeling using the UML: a Case Study". In: *Journal of Networks* 7.1 (2012), p. 4.

[117] J. E. Van Aken. "Management research as a design science: Articulating the research products of mode 2 knowledge production in management". In: *British journal of management* 16.1 (2005), pp. 19–36.

[118] A. Van Lamsweerde. "From system goals to software architecture". In: *Formal Methods for Software Architectures* (2003), pp. 25–43.

[119] A. Van Lamsweerde. "From system goals to software architecture". In: *Formal Methods for Software Architectures*. Springer, 2003, pp. 25–43.

[120] J. Van't Wout, M. Waage, H. Hartman, M. Stahlecker, and A. Hofman. *The integrated architecture framework explained: why, what, how*. Springer Science & Business Media, 2010.

[121] S. L. Vargo, P. P. Maglio, and M. A. Akaka. "On value and value co-creation: A service systems and service logic perspective". In: *European management journal* 26.3 (2008), pp. 145–152.

[122] N. Venkatraman. "Beyond outsourcing: managing IT resources as a value center". In: *MIT Sloan Management Review* 38.3 (1997), p. 51.

[123] E. Walters. *Modeling the Business Model Canvas with the ArchiMate specification*. `https://publications.opengroup.org/w195`. 2020.

[124] C. Weekly. *Case studies; The sytems behind the shows*. Internet. 2006.

[125] R. J. Wieringa. "Design Science as Nested Problem Solving". In: *Proceedings of the 4th International Conference on Design Science Research in Information Systems and Technology, Philadelphia*. Philadelphia: ACM, 2009, pp. 1–12.

[126] R. Wieringa, W. Engelsman, J. Gordijn, and D. Ionita. "A business ecosystem architecture modeling framework". In: *2019 IEEE 21st Conference on Business Informatics (CBI)*. Vol. 1. IEEE. 2019, pp. 147–156.

[127] R. J. Wieringa. *Design science methodology for information systems and software engineering*. Springer, 2014.

[128] E. Yu, M. Strohmaier, and X. Deng. "Exploring intentional modeling and analysis for enterprise architecture". In: *Enterprise Distributed Object Computing Conference Workshops, 2006. EDOCW'06. 10th IEEE International*. IEEE. 2006, p. 32. ISBN: 0769527434.

[129] E. Yu. “Towards modelling and reasoning support for early-phase requirements engineering”. In: *Requirements Engineering, 1997., Proceedings of the Third IEEE International Symposium on.* IEEE Computer Society Press, 2002, pp. 226–235. ISBN: 0818677406.

[130] E. Yu and J. Mylopoulos. “Towards modeling strategic actor relationships for information systems development — with examples from business process reengineering”. In: *Proceedings of the 4th Workshop on Information Technolgies and Systems (WITS’94).* 1994.

[131] J. A. Zachman. *The concise definition of the Zachman framework.* `https://www.zachman.com/about-the-zachman-framework`. 2017.

An enterprise architecture (EA) is a high-level representation of the enterprise. An EA is designed to realize the business value of an organization. In the first part of this thesis a traceability relation between the EA, modeled with ArchiMate, and the business goals is defined and empirically evaluated. Afterwards the language is validated with practitioners and academics in terms of understandability. The conclusion is that the concepts of this modeling language are difficult to understand because the concepts are too closely related. We end this part with a simplification of the goal-modeling language.

In the second part a traceability relation between two conceptual modeling languages, e3value and ArchiMate, is designed. This traceability relation is refined in an experiment with practitioners and evaluated in a case study. A key finding from this case study is that for the traceability to be useful, the quantifications of an e3value model have to be aligned with those in an ArchiMate model. The thesis ends with a quantitative alignment of an ArchiMate model with the quantifications of an e3value model.

ISBN: 978-90-365-5320-9