Softw Syst Model
DOI 10.1007/s10270-012-0299-z

THEME SECTION PAPER

Bridging value modelling to ArchiMate via transaction modelling

Sybren de Kinderen · Khaled Gaaloul · Henderik A. Proper

Received: 1 November 2011 / Revised: 14 September 2012 / Accepted: 25 October 2012

Abstract The ArchiMate modelling language provides a coherent and a holistic view of an enterprise in terms of its products, services, business processes, actors, business units, software applications and more. Yet, ArchiMate currently lacks (1) expressivity in modelling an enterprise from a value exchange perspective, and (2) rigour and guidelines in modelling business processes that realize the transactions relevant from a value perspective. To address these issues, we show how to connect e^3value, a technique for value modelling, to ArchiMate via transaction patterns from the DEMO methodology. Using ontology alignment techniques, we show a transformation between the meta models underlying e^3value, DEMO and ArchiMate. Furthermore, we present a step-wise approach that shows how this model transformation is achieved and, in doing so, we also show the relevance of such a transformation. We exemplify the transformation of DEMO and e^3value into ArchiMate by means of a case study in the insurance industry. As a proof of concept, we present a software tool supporting our transformation approach. Finally, we discuss the functionalities and limitations of our approach; thereby, we analyze its advantages and practical applicability.

Keywords ArchiMate · e^3value · DEMO · Meta model · Model transformation.

Called Erik by family and friends, but known as Henderik in his passport and to the *h*-index.

S. de Kinderen (✉) · K. Gaaloul · H. A. Proper
Public Research Centre Henri Tudor,
1855 Luxembourg-Kirchberg, Luxembourg
e-mail: sybren.dekinderen@tudor.lu

H. A. Proper
iCIS, Radboud University Nijmegen,
6500 GL Nijmegen, The Netherlands

1 Introduction

ArchiMate is an Open Group standard [17,27] for the modelling of enterprise architectures,[1] emphasizing a *holistic* view of the enterprise. This means that architects can use ArchiMate to model, amongst others, an organization's products and services, how these products and services are realized/delivered by business processes, and how in turn, these processes are supported by information systems and their underlying IT infrastructure. Such a holistic perspective on an enterprise helps to guide change processes [19], provides insight into cost structures [3], and more [27].

Because of its inherent holistic nature, ArchiMate lacks specific guidelines for modelling an enterprise from a value exchange perspective [31]. A value perspective focuses on depicting the value exchanges between actors participating in a value web, describing what each actor offers to others, and what it receives in return. For example, an online music store ships an 'LP' to the customer, and receives 'Money' as compensation. Such a perspective would nicely complement ArchiMate in the sense of providing an economic rationale, in terms of value exchanges, for the largely operational information, such as business processes and IT infrastructure, expressed in an ArchiMate model [3].

To address the lacking of a value perspective in ArchiMate, we explored in earlier work [7] a formal transformation of the well-established value modelling technique e^3value into ArchiMate. On the one hand, we found there to be a conceptual overlap that indeed allows us to create a formal transformation between these techniques. However, on the other hand, we found that e^3value and ArchiMate differ substantially in the level of abstraction of the information expressed in these models. For example, whereas e^3value

[1] http://www.opengroup.org/archimate/.

Published online: 20 November 2012

focuses on the high-level value exchanges 'LP' for 'Money', ArchiMate focuses directly on the detailed realization of this exchange, in terms of 'A delivery of the LP to the customer' (a business process) and 'Track-and-trace' (the supporting IT infrastructure). In other words, there exists a gap between economic transactions modelled with e^3value and the underlying business processes and IT infrastructure modelled with ArchiMate. Ideally, this gap should also be filled by a formal method.

In this paper, we present a formal transformation of e^3value into ArchiMate via DEMO. DEMO, short for Design and Engineering Methodology for Organizations, is a method comprising of a comprehensive set of conceptual modelling techniques, in combination with a theory-based way of thinking and associated way of working, focused on modelling/analyzing/designing the *essential* aspects of an organization [11,12]. DEMO uses the word *essential* here to refer to the implementation-independent aspects of an organization. As such, DEMO aims to abstract away from implementation-specific details, such as the information systems present in a business collaboration.

Using DEMO as an intermediate between e^3value and ArchiMate would also enable architects to use the semantically rich way of thinking of DEMO to create ArchiMate models starting from the economic transactions modelled in e^3value. These DEMO models would then primarily be ArchiMate models providing an essential view of the business processes (business layer) and the information processing (application layer) in the enterprise. In sum, we connect e^3value, DEMO and ArchiMate because (1) e^3value provides an economic rationale for a business operationalization in ArchiMate, while (2) DEMO offers a bridge between e^3value and ArchiMate, providing strict guidance for translating economic transactions into their operationalization.

While ArchiMate, e^3value and DEMO are the result of academic research, they have all demonstrated a clear impact in practice. ArchiMate has evolved to become The Open Group's standard for enterprise architecture description [17]. e^3value and DEMO have not yet evolved to become internationally accepted standards. However, both have indeed proven their use in practice by amongst others, (1) assessing partnerships for a merger in the banking industry (e^3value) [23], (2) acting as a basis for value-based matchmaking between needs and services in the healthcare industry (e^3value) [2], (3) acting, in a government agency, as a point of departure for business process modelling (DEMO) [30], and, by (4) fostering, in the aerospace industry, a shared understanding of fragmented strategic concerns, and a link of these strategic concerns to design principles (DEMO) [20].

Our contribution is a formal transformation of e^3value to ArchiMate via DEMO, in terms of (1) a formal mapping of the meta models underlying these techniques, and (2) a systematic application of these meta models to map a model created in e^3value to a model of an enterprise architecture in ArchiMate. In addition, we discuss a software tool that acts as a proof-of-concept for the formal mapping between these techniques. Moreover, we discuss why the particular chaining of e^3value, DEMO and ArchiMate makes sense from a pragmatic point of view.

Note that parts of this paper have already been published earlier, in [7] and [8]. However, [7] only *explores* a transformation between e^3value and ArchiMate, without providing a formal transformation between these techniques (as in this article), nor does it explore the use of DEMO as a bridge between e^3value and ArchiMate. On the other hand, [8] provides a formal transformation of DEMO and ArchiMate. It thereby explains only *a part of* the modelling chain that is presented in this paper. As such, compared to earlier work, this paper provides a full *formal* presentation of the transformation of e^3value to ArchiMate via DEMO, including a rationale of this particular chaining of techniques.

We use a running example of an insurance scenario to illustrate our ideas. In addition, as a (partial) computational assessment, we discuss a tool implementation of the formal transformation between DEMO and ArchiMate. Moreover, we analyze the advantages of our transformation approach and its practical applicability.

The remainder of this paper is organized as follows. Section 2 introduces the insurance case study, and (briefly) discusses e^3value. In Sect. 3, we discuss the transformation of e^3value to ArchiMate, and how to use DEMO as a bridge. Subsequently, we discuss in detail how to, firstly, map e^3value to DEMO (in Sect. 4), and, secondly, how to map DEMO to ArchiMate (Sect. 5). Section 6 discusses tool support for our model transformation. Section 7 provides a discussion of our approach, in terms of (1) limitations imposed by the use of DEMO, and (2) the planned practical validation of our approach. Section 8 presents related work. Finally, Sect. 9 concludes and discusses directions for further research.

2 The Servinsurance value web

For illustration purposes, we present a fictitious but realistic insurance case. This case is inspired by a paper on the economic functions of insurance intermediaries [4], as well as the running case used to illustrate the ArchiMate language specification [18,25,26].

Servinsurance, a large insurance company, offers one of its many products, car insurance, to the customer via insurance brokers. The main reason for selling insurance via brokers is to reduce the risk of *adverse risk profiles* [4], incomplete or faulty risk profiles of customers that lead insurance companies to sell inappropriate insurance packages. To mitigate adverse risk profiles, insurance companies may therefore

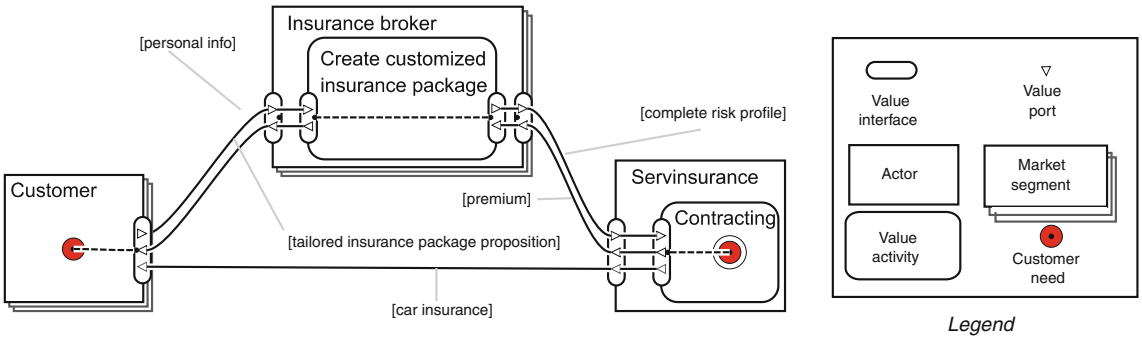

Fig. 1 Value model of selling car insurance via an intermediary

rely on insurance brokers, whose core business is to match customer profiles to appropriate packages.

We use e^3value [14] to model the Servinsurance intermediary sales model. e^3value focuses on modelling the value exchanges between actors participating in a value web, depicting what each actor offers to others, and what it receives in return. The principle of economic reciprocity, which states that an offering from an actor should always be compensated, is central to e^3value; in other words, one good turn deserves another.

While e^3value provides a lightweight, visual, modelling technique for discussing alternative value webs with persons with a business background (typically in a workshop-like setting), it is nevertheless firmly grounded in *computer science*. e^3value is based on a meta model (see Fig. 3) and provides computational support for creating value models by means of a software tool that is based on the e^3value meta model.

The e^3value model for the intermediary sales model is depicted in Fig. 1. Here, we see that:

- The customer provides 'personal information' (a *value object*) to the intermediary, which is valuable to the latter because it allows for the composition of a complete risk profile. The provisioning of 'personal information' by the customer is modelled by an outgoing *value port*, whereas the reception thereof by the insurance intermediary is modelled by an incoming port.
- Using 'Create customized insurance package' (a *value activity*), the intermediary matches customer information to appropriate offerings, and possibly requests additional customer information. Note that the value model shows only the (high level) value-adding activities, not the underlying operational business processes realizing these activities.
- The intermediary provides Servinsurance with a 'Complete risk profile' (a *value object*), which, as discussed, the insurer may use to mitigate adverse selection of risk profiles.
- To compensate for 'Complete risk profile', Servinsurance pays the intermediary a 'Premium'. This compensation is modelled by grouping, by means of *a value interface*, value ports to which the value objects 'Complete risk profile' and 'Premium' are attached.
- The customer receives a 'Tailored insurance package proposition' from the intermediary, and the actual 'Car insurance' from Servinsurance in return. This is to depict the advantage that, through intervention of the intermediary, the customer receives an insurance package in line with his profile, against an appropriate fee.[2]

3 Transforming e^3value to ArchiMate, using DEMO as a transformation engine

We aim to transform e^3value to ArchiMate via DEMO.

ArchiMate provides a language for modelling an organization's enterprise architecture. As such, it aims to express an organization holistically.

Yet, ArchiMate lacks expressivity for modelling an enterprise from a value perspective. In particular, while there is a 'value' concept in ArchiMate, considerations such as financial independence, a consistent interpretation of the notion of value, and economic reciprocity, which are standard to the value modelling technique e^3value, are missing in ArchiMate. This lacking of an extensive value perspective from ArchiMate is logical given that the primary aim of ArchiMate is to provide a concise, A3-sized, holistic model of the organization. By design, ArchiMate therefore does not fully integrate constructs from different specialized languages, such as parallelization from process modelling languages, access rights from security languages, or economic reciprocity from value modelling languages. This is because such full integration would likely lead to 'modelling spaghetti': cluttered models, whereby one loses track of the overall, holistic, picture of an organization.

[2] Note that 'fee' is not included in the value model, because the value model depicts only the initial acquisition of an insurance package, not the subsequent monthly compensation thereof.

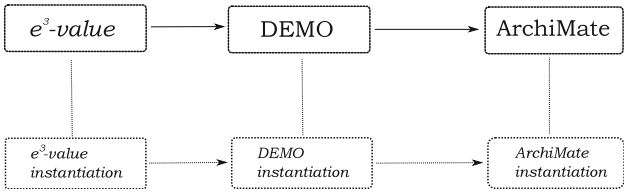

Fig. 2 Transforming a value (meta) model to an ArchiMate (meta) model via a DEMO transaction (meta) model. Adapted from [5,29]

Nevertheless, as discussed in [3,7], ArchiMate would benefit from a formal linkage to e^3*value*. On the one hand, ArchiMate complements e^3*value* in terms of operationalizing a business collaboration in terms of the underlying business processes and information systems. On the other hand, e^3*value* complements ArchiMate in terms of providing profitability calculations for a given business operationalization modelled in ArchiMate.

However, a direct transformation from e^3*value* to ArchiMate is inhibited by the difference in level of abstraction between the economic transactions modelled in e^3*value* and the business processes and IT infrastructure modelled in ArchiMate [7]. In particular, we would lack formal guidance in creating process models that realize the economic transactions from e^3*value*. To address this issue, we use DEMO and in particular its transaction patterns as an intermediate between e^3*value* and ArchiMate.

In model transformation terms [5,29], DEMO acts as a *transformation engine* between e^3*value* and ArchiMate. It specifies the transformation rules necessary to bridge the gap between an e^3*value* value model and an operational ArchiMate process model. DEMO can act as such a transformation engine because of its process-based patterns that describe, in an organization-independent way, how an economic transaction should be realized in terms of a detailed business process. We describe the core ideas behind DEMO, and the DEMO transaction patterns, in further detail in the next section.

We depict our transformation of e^3*value* to ArchiMate via DEMO in Fig. 2. Here, the top layer depicts the transformation of the e^3*value* and ArchiMate meta models via the DEMO meta model, while the lower layer depicts that transformations of *an instantiation* of the e^3*value* model and ArchiMate models requires an instantiation of the DEMO meta model as well. In particular, we require an instantiation of the DEMO transaction pattern to help us to translate economic transactions into business processes.

4 Specifying value exchanges from e^3*value* in DEMO transaction models

Now, we detail the formal transformation depicted in Fig. 2: e^3*value* to DEMO in Sect. 4, and DEMO to ArchiMate in Sect. 5.

4.1 The DEMO meta model and transaction pattern

As stated in the introduction, DEMO is a comprehensive set of conceptual modelling techniques focused on modelling the *ontological* aspects of an organization [11,12].

DEMO achieves its focus on ontological aspects by perceiving of an organization as a *social* system of actors, that collaborate to achieve a common goal. Chief to this collaboration are acts: production acts, and communication acts. Production acts bring about (part of) a good or service, and directly contribute to achieving the organization's common goal. In the Servinsurance case, a production act is, for example, 'Find matching insurance package', as executed by the insurance broker on behalf of the customer. Communication acts, then, serve to coordinate among the actors that either receive results from, or execute, the production acts. In the Servinsurance case, 'Apply for insurance' is, for example, a communicative act used by the customer to indicate to the insurance broker the interest in an insurance package.

In this paper, we use only that subset of the DEMO conceptualization and techniques that are relevant to our purposes: to plug the gap between high-level, economic, transactions stemming from e^3*value* and detailed ArchiMate models of organization specific business processes and information systems. In particular we require (1) a subset of the DEMO meta model, depicted in Fig. 3, with an explanation of the DEMO concepts in Table 1, and (2) the DEMO transaction pattern.

The DEMO standard transaction pattern focuses on a process-based pattern of (instantiations of) DEMO meta model concepts, showing the sequence of acts that *always* needs to be executed to realize an economic transaction. Therefore, here we see again DEMO's emphasis on the ontological aspect of an organization: no matter what the domain, if we perceive of an organization as a social entity, then we see a pattern of generic acts that *always* occurs in carrying out a transaction [11]. For example, one actor always has to initiate a transaction by performing the act 'Request' (which in the Servinsurance case may translate to the act 'Apply for insurance' as carried out by a customer), while another actor always has to perform the 'Execute' act in order to produce the good or service that the initiating actor is interested in (in the Servinsurance case, this may translate to the act 'Find matching package' which is executed by the insurance broker. Note that 'Find matching package' is considered as an execute act because it reflects the core matchmaking capability that an insurance broker provides.). Note, finally, that the DEMO standard transaction pattern is grounded in the well-established Language/Action Perspective (LAP) on information systems [36], wherein patterns of acts describe the 'conversational dance' (cf. [36]) that needs to be carried out between actors to execute an (economic)transaction.

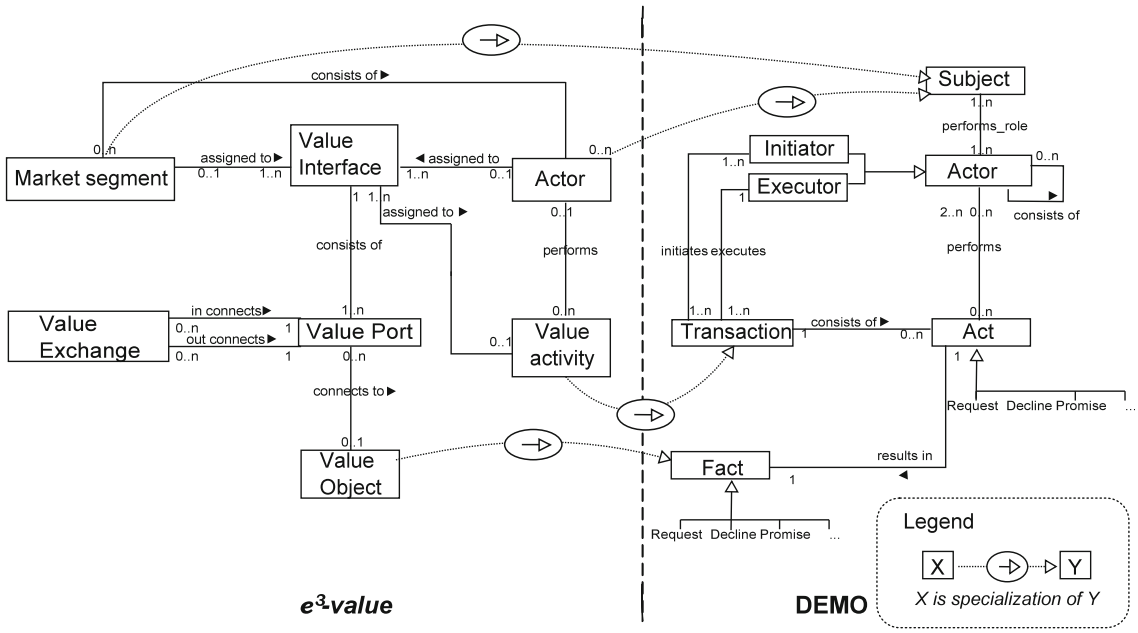

Fig. 3 Mapping of e^3*value* and DEMO meta models

Table 1 Definition of DEMO Meta model concepts, adapted from [12]

DEMO concept	Definition	Servinsurance example
Subject	A real world entity (human, business unit, or otherwise) that has a social role to play as an actor in an organization	Servinsurance
Actor	A social role played by an entity in an organization	The actor 'Insurer', played by the subject 'Servinsurance'
Act	An act is performed by an actor, to either (1) contribute to bringing about a good or service, or (2) coordinate part of a transaction to bring about a good or service	'Find matching package' (carried out by the insurance broker)
Transaction	A collection of acts, defined according to a pre-specified DEMO transaction pattern	The transaction 'Create customized insurance package', as executed by the actor 'Insurance broker'. Note that part of this transaction is realized by the act 'Find matching package' (by the insurance broker) but also by the act 'apply for insurance', which is carried out by the insurance customer to trigger the set of acts that collectively make up this transaction
Fact	The result of an act	The fact 'Matched package', as a result of the act 'Find matching package'

4.2 Mapping the e^3*value* meta model to the DEMO meta model

Now that we have introduced the DEMO meta model and transaction pattern, we translate an e^3*value* model into a DEMO process model. We do this in three steps:

1. Translate the concepts from an e^3*value* value model into DEMO concepts (in Sect. 4.2.2).
2. Create a high-level DEMO transaction model, capitalizing on the meta model mapping (in Sect. 4.2.3). As implied by name, this model shows transactions only, and the actors involved in them.
3. Apply the DEMO standard transaction pattern to create DEMO process models (in Sect. 4.2.4).

However, before discussing the transformation in detail, we first discuss the used mapping technique.

4.2.1 Mapping technique used for model transformation

For mapping *e*^{3}*value* to DEMO and, later on, mapping DEMO to ArchiMate, we use the meta model mapping technique described in [37]. This mapping technique distinguishes different types of mappings, the most relevant for our work being (1) *class-to-class mappings*, which relate a concept from meta model A (e.g. an Actor from *e*^{3}*value*) to a concept from meta model B (e.g. an 'Actor' from *e*^{3}*value* relates to a 'Subject' from DEMO). and (2) *relation-to-relation mappings*, which relate concept relationships from meta model A (e.g. the DEMO relation 'performs_role' between a Subject and an Actor) with concept relationships from meta model B (e.g. 'performs_role' between the concepts Subject and Actor from DEMO relates to the ArchiMate relation 'assigned_to' between the concepts Actor and Business role). Furthermore, [37] distinguishes between different types of mapping relations, the most important for us being: equivalence, is generalization of, and its inverse is specialization of, and no relation. Finally, note that [37] grounds its meta model mappings in well-established ontology-alignment mapping relations. For example, its semantic mapping relations have been applied in the context of online ontologies in [22].

From [37] we use only the concept mappings described above. We do not use its integration rules since these are aimed at how to *integrate* two conceptual models, while we define a *transformation* between models.

Instead of integration rules, we rely on transformation rules [5, p. 627]. Transformation rules implement concept mappings by providing an injection from a left-hand side (for example, an *e*^{3}*value* actor) to a right-hand side (for example, a DEMO subject), and can be defined in model transformation software tools. Subsequently, using the transformation rules, these software tools can formally transform one meta model instantiation into another meta model instantiation. We will show an example of this for our model transformation chain in Sect. 6, where we discuss our software tool.

Note that, in line with our transformation chain depicted in Fig. 2, our transformation rules are unidirectional, meaning that we transform *e*^{3}*value* to DEMO to ArchiMate, but not vice versa.

Using the presented mapping technique, we can now transform *e*^{3}*value* models into DEMO models.

4.2.2 Step 1: Translate concepts from e^{3}value into DEMO concepts

We present our *e*^{3}*value*—DEMO meta model mapping in Fig. 3, with a mapping rationale in Table 2. For the Servinsurance case, this mapping provides us the following results:

- Actors and market segments from *e*^{3}*value* map to subjects in DEMO. We map the market segments 'Customer' and 'Insurance broker' to the subjects 'Customer' and 'Insurance broker', and map the actor 'Servinsurance' to the subject 'Servinsurance' in DEMO.
- Value activities from *e*^{3}*value* map to transactions in DEMO. We map the value activities 'Create customized insurance package' and 'Contracting' to DEMO transactions.
- Value objects from *e*^{3}*value* map to facts from DEMO. We map the value objects 'Tailored insurance package proposition', 'Car insurance', 'Personal info', 'Premium', and 'Complete risk profile' to DEMO facts.

Observe that we did not map any concept-to-concept *relations*. This is because of the focus on the 'Actor' concept in DEMO (see Fig. 3), implying that most DEMO concepts, such as 'Transaction' and 'Act', are related to roles and not the real-world entities that fulfill this role. Yet, the DEMO concept of an 'Actor' has no simulacrum in *e*^{3}*value*. After all, *e*^{3}*value* actors focus *on real-world entities* (or collections thereof, such as market segments), whereas DEMO actors focus on *the role* that these real-world entities play in the organization at hand. As a result, we cannot inherit, from *e*^{3}*value* to DEMO, relations between concepts, and have to re-create these relations in DEMO after having inherited the concepts from *e*^{3}*value*. This essentially means that we have to deconstruct the *e*^{3}*value* model and, later, reconstruct the transaction model, simply because of one concept mismatch.

Note that unmapped concepts and relations, such as 'value port' (see Table 2), are not used further in the transformed DEMO models (and, thus, they also do not become part of the ArchiMate models). This is because they have no counterpart in the target model. Since we leave the meta models themselves untouched, such concepts and relations are not transformed.

4.2.3 Step 2: Mapping an e^{3}value value model to a DEMO transaction model

In this step, we create a DEMO transaction model, taking as input the *e*^{3}*value* concepts that DEMO inherited in the previous step. The DEMO transaction model serves as input for creating detailed process models in the next step.

For the Servinsurance case, the transaction model can be found in Fig. 4. We create this transaction model as follows:

1. *Define the actors personified by the subjects.* For example: Servinsurance plays the actor role of an 'Insurer'.
2. *Link transactions to actors.* Because, as stated, we now only have loose concepts (i.e., the subjects, and transactions, but without any relation between them), we have to use *e*^{3}*value* as a frame of reference to link these concepts

Table 2 Explaining the conceptual mapping between e^3*value* and DEMO in Fig. 3

e^3*value*	DEMO	Mapping rationale for concepts
Actor/Market Segment	Subject	An e^3*value* actor is a profit-and-loss responsible real-world entity, thus specializing a DEMO subject, which is real-world entity. Observe that an e^3*value* actor does *not* map to a DEMO actor: in DEMO, an actor refers to *a role* played by a real-world entity, not the real-world entity itself. As a collection of e^3*value* actors, A e^3*value* market segment also specializes a DEMO subject, most prominently because they fulfill no specific role (after all: the actors in a market segment only define a value object in the same way)
Value object	Fact	Facts, as social counterparts to value objects, encompass valuable objects exchanged between actors, supplemented with facts necessary from a social perspective (e.g. in the case of Servinsurance: a fact by which the customer indicates interest in an insurance package)
Value activity	Transaction	An e^3*value* value activity shows the high-level value adding activities of an organization, which act as a starting point for DEMO transactions, the transactions being specified in terms of business processes in which the more elementary acts are performed (that together, on a more operational level, realize DEMO transactions)

e^3*value*–DEMO, no relation

The e^3*value* concepts *value interface, value port, value exchange* are specific to e^3*value* and have no related DEMO concepts. The value port is specific to e^3*value* because it allows the modeller to concentrate on value exchanges between actors, rather than on the (internal) business processes realizing the value exchanges. The concepts 'Value Interface' and 'Value Exchange' are specific to e^3*value* because they focus on showing the collection of *value* objects exchanged: the value interface by grouping value ports together, the value exchange to refer to a collection of value objects that is exchanged together

together. We do this in four steps (1) Tracing an actor to a subject. For example, in the DEMO transaction model (Fig. 4), the actor 'Insurance broker' relates to the subject 'Insurance broker', (2) Mapping the DEMO subject back to e^3*value*. For example, the subject 'Insurance broker' maps to the market segment 'Insurance broker'. (3) Finding the value activity executed by the e^3*value* actor or market segment. For example, the market segment 'Insurance broker' executes the value activity 'Create customized insurance package'. And finally, (4) finding the actor or market segment receiving results from the value activity. For example, the market segment 'Customer' receives results from the value activity 'Create customized insurance package'.

3. *Identify, for each transaction, possible elementary roles, and the subjects fulfilling these roles.* We find the elementary roles that carry out a transaction. This means that we analyze what the most elementary subjects and actors are that have a social part to play in carrying out a transaction. For example, in the transaction model of the Servinsurance case (Fig. 4), we see that, within the insurer Servinsurance, we have a 'car insurance' department that, in the actor role of an 'underwriter', is actually responsible for carrying out the transaction 'Contracting'. Note that one does not model the department 'Car insurance' in e^3*value*. This is because it is not profit-and-loss responsible.

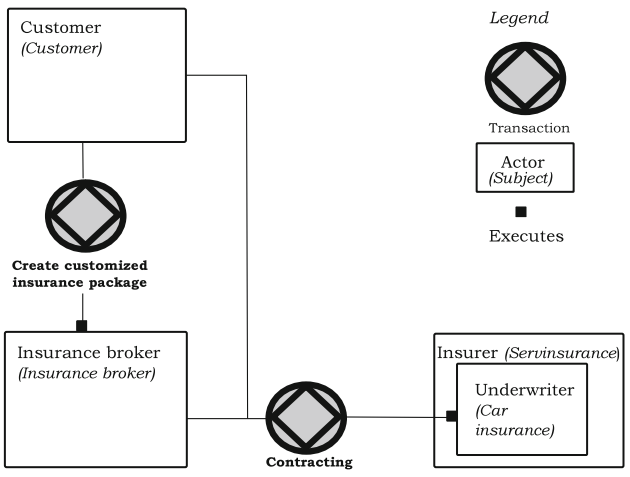

Fig. 4 The DEMO transaction model

4.2.4 Step 3: Apply the DEMO standard transaction pattern to create DEMO process models

In this step, we create a DEMO process model from the DEMO transaction model defined thus far, by applying to the DEMO transaction model the DEMO standard transaction pattern.

As stated in Sect. 4.1, the standard transaction pattern describes the 'conversational dance' carried out between two or more actors involved in a transaction. Essentially, such a conversational dance describes the pattern of illocutionary acts that is always carried out between *two types of actors* that together realize a transaction: *the initiator*, who initiates a transaction, and often receives results from it in terms of a good or service, and *the executor*, who brings about part of the good or service on behalf of the initiator.

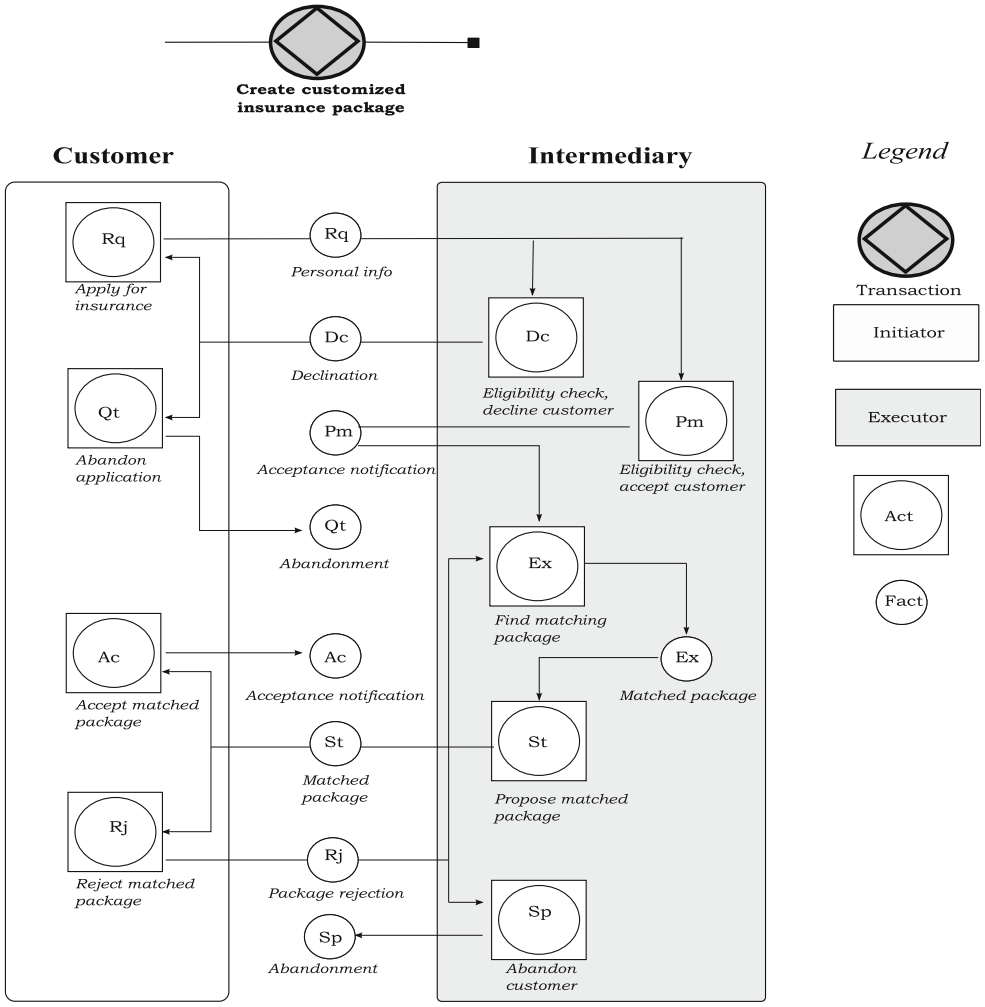

Fig. 5 The DEMO process model

For example, for detailing the Servinsurance transaction 'Contracting' in Fig. 5, we apply the acts from the DEMO transaction pattern as follows:

- an initiator that performs a request act, namely the insurance broker sending an insurance request to the underwriter;
- an executor that performs an execute act, namely the underwriter that underwrites the insurance package using the customer risk profile (underwriting means that the premium for the requested package is calculated based upon the customer's risk profile).
- and an initiator that performs an accept act, namely the customer accepting the insurance policy.

Finally, note that we can now also use the DEMO facts that we have inherited in step one from e^3value. For example, we can place the fact 'personal info' as an outcome of the act 'apply for insurance' (as performed by the customer), and the fact 'complete risk profile' as an outcome of the act 'send insurance request' (as performed by the insurance broker). Obviously, we supplement these facts inherited from e^3value with facts necessary from a social perspective, i.e., facts necessary in the communication between actors, such as 'acceptance notification'.

Note that DEMO also has more elaborate transaction patterns that expand upon the standard transaction patterns. Most prominently, these detail patterns of communicative acts and facts for choices. However, for the sake of clarity, we choose to focus on showing the high-level process generated by the DEMO transaction patterns.

5 Translating DEMO process models to ArchiMate

In this section, we first introduce the ArchiMate business layer meta model. Thereafter, we present the mapping between the DEMO and the ArchiMate business layer meta models. Subsequently, we apply this mapping to transform the DEMO process model of Servinsurance into an

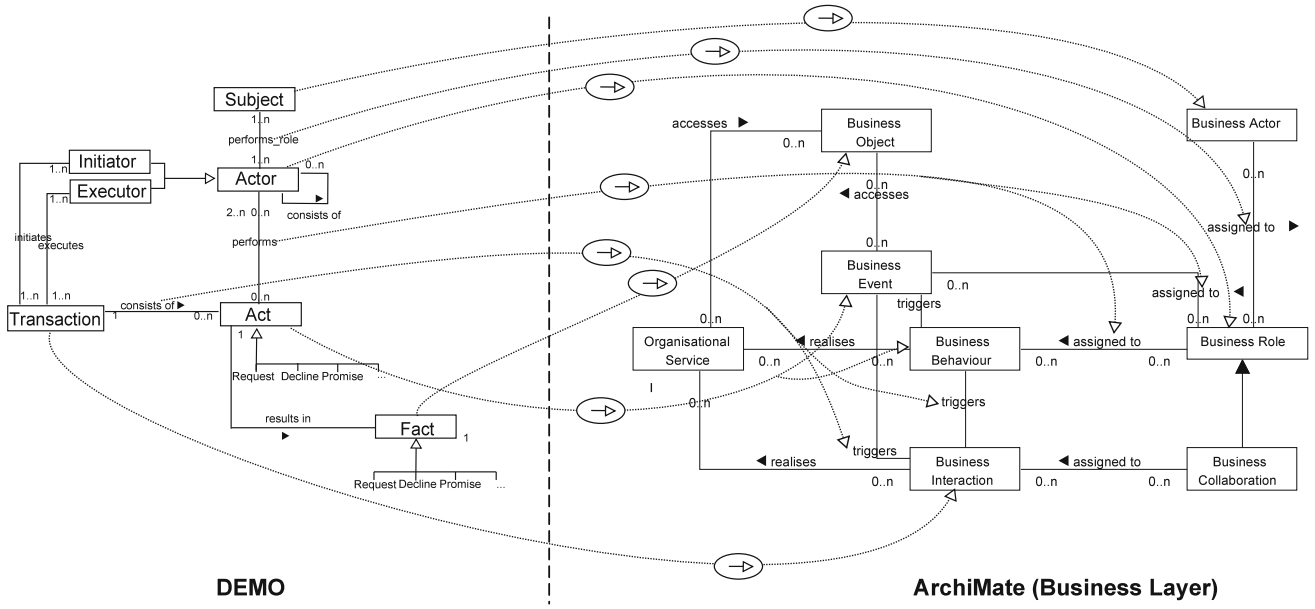

Fig. 6 Mapping of DEMO and ArchiMate meta models

ArchiMate model. The idea is to leverage the operationalization of a business process modelled in DEMO by means of modelling, in ArchiMate, the underlying applications and the IT infrastructure.

5.1 The ArchiMate business layer meta model

Figure 6 provides an excerpt of the ArchiMate business layer concepts and their relations. As implied by name, the business layer focuses on an organization's business concepts such as products, (commercial) services, and business processes. A description of the main concepts and a link to the Servinsurance case study are detailed in Table 3. Note that we use only an excerpt of the ArchiMate business layer meta model so as to focus on those concepts and relations that can be mapped from DEMO.

5.2 Mapping the DEMO meta model to the ArchiMate meta model

Now that we have presented an excerpt of the ArchiMate business layer meta model, we translate a DEMO process model into an ArchiMate business layer model.

5.2.1 Step 1: Transforming DEMO concepts to ArchiMate concepts

In this step, we apply the DEMO–ArchiMate meta model mapping from Fig. 6, and the corresponding rationale of our meta model mapping (i.e., Table 4). In Fig. 6, we define a specialization relation between the mapped concepts from DEMO to ArchiMate. Note that here, as in Sect. 4.2.2, we rely again upon the mapping technique described in Sect. 4.2.1.

For Servinsurance, the mapping is exemplified in Table 4 (mapped concepts) and Table 5 (mapped relations). For reference, see the Servinsurance ArchiMate model in Fig. 7, and the Servinsurance DEMO process model in Fig. 5.

5.2.2 Step 2: Define an enterprise architecture model from the mapped concepts

From our mapping, we now create an enterprise architecture model in ArchiMate in two steps: (1) *Create a process model from the mapped DEMO concepts and relations.* For example, for the Servinsurance case, Fig. 7 details exactly what *operational* business process steps realize the business interaction 'Create customized insurance package': the steps 'Apply for insurance', 'Eligibility check', 'Find matching package', 'Propose matched package', 'Accept matched package' and 'Send insurance request'.

(2) *Create a model of the IT infrastructure supporting the business process*. Using ArchiMate modelling, we now describe the IT infrastructure supporting the modelled business process. For example: for the Servinsurance case, Fig. 7 details the support of the application service 'Risk assessment service' for the business process steps 'Eligibility check' and 'Underwrite insurance'. Note here that aforementioned example illustrates the link that ArchiMate establishes between IT-infrastructure and business processes.

Table 3 ArchiMate business layer meta model concepts, adapted from [17,28]

ArchiMate concepts	Definition	Servinsurance
Business actor	Individual persons (e.g., customers or employees), but also groups of people (e.g., departments or business units) within the organizations	The actor 'Servinsurance' is the business actor in this case
Business role	A role that an actor fulfills in an organization. Importantly, this role is usually defined as the work carried out by an actor [28]	The role 'Insurer', played by the actor 'Servinsurance'
Business collaboration	"A (temporary) configuration of two or more business roles resulting in specific collective behavior in a particular context." [17]	This concept is not represented in our scenario, but is required for the concept 'Business interaction'
Organizational service	"A unit of functionality that is meaningful from the point of view of the environment" [17]. The following concepts realize a service [17]: *Business processes, business functions, business interactions*. Moreover, A business process/function is "a unit of internal behaviour, performed by one or more roles within the organization " [17]. Finally A business interaction is "a unit of behaviour similar to a business process or function, but it is performed in a collaboration of two or more roles within the organization" [17]	'Car insurance service' defines an organizational service where 'Contracting' is the business function and 'Register customer profile' is a business process
Business event	According to [17], a business event is "Something that happens (externally) and may influence business processes, functions or interactions. A business event is most commonly used to model something that triggers behaviour, but other types of events are also conceivable: e.g., an event that interrupts a process."	The business event 'Find matching package'
Business object	An entity that "is manipulated by behaviour such as business processes or functions" [17]	A 'Contract' document is a business object in the Servinsurance process

Table 4 DEMO—ArchiMate meta model concepts mapping

DEMO/ArchiMate	Mapping rationale for concepts	Example transformation
Subject/Business actor	A subject is an actor in an organization, which corresponds to the definition of a business actor in ArchiMate	The subject 'car insurance' (a department) to the business actor 'car insurance'
Actor/Business role	An actor in DEMO refers to a social role played by a subject in an organization. Such a social role corresponds to the definition of a business role in ArchiMate where roles are typically used to distinguish responsibilities	The actor 'Underwriter' to the business role 'Underwriter'
Act/Business behaviour or event	An act is performed by a subject member of a social role. Its scope is about contribution/coordination for services. In the ArchiMate context, it corresponds to the realization of an organizational service via a business process or a function (business behaviour) or a business event (e.g., external request)	The act 'Find matching package' to the business event 'Find matching package'
Transaction/Business interaction	In DEMO, transactions are always initiated and executed by different roles. This emphasizes the interaction aspect that we can find in ArchiMate, where a business interaction requires more than one role to perform an organizational service	The transaction 'Create customized insurance package' to the business interaction 'Create customized insurance package'
Fact/Business object	A fact is the result of an act. In other words, it represents a product which is from an ArchiMate perspective an element accessing a business object and encapsulated within an organizational service	The fact 'Tailored insurance package proposition' to 'Contract file' in Archimate

Table 5 DEMO—ArchiMate meta model relations mapping

DEMO/ArchiMate	Mapping rationale of relations	Example transformation
Performs_role/assigned_to	In both DEMO and ArchiMate, one relates a real world entity (e.g., Servinsurance) to a role played by that entity (e.g., the role of insurer in the case of Servinsurance)	The department 'Car insurance' performs the role of 'Underwriter'
Consists_of/triggers	As transactions map to business interactions, and acts map to business events and business behaviour, the relation 'consists_of' between transactions and acts in DEMO maps logically to the relation 'triggers' between business interactions and business events/business behaviour in ArchiMate	'Create customized insurance package' consists of (the more detailed) 'apply for insurance' and 'find matching package'
Performs/assigned_to	While DEMO and ArchiMate use different nomenclature in both a role—not the real-world entity behind it—carries out acts	The underwriter carries out the act 'Underwrite insurance'

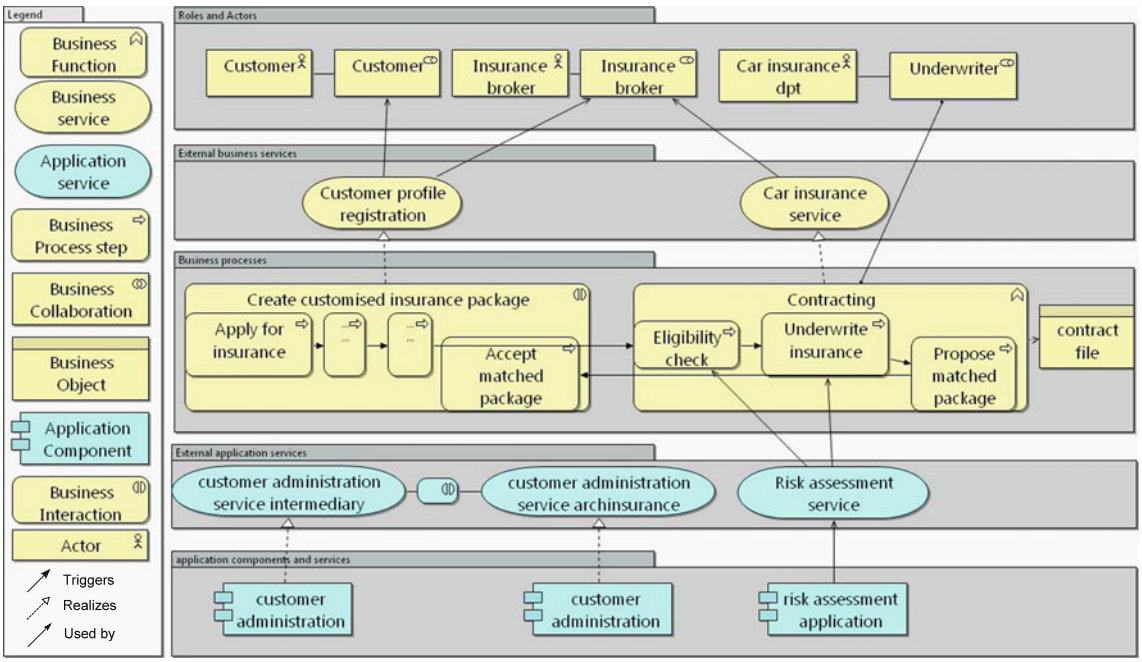

Fig. 7 (Partial) enterprise architecture model, based on DEMO

6 Computational assessment

To ensure a (partial) computational assessment of the mapping discussed in this paper, we have implemented the DEMO to ArchiMate mapping (summarized in Tables 4 and 5) in ATL,[3] an Integrated Development Environment for implementing model transformations that is built on top of the Eclipse platform. This mapping conforms exactly to the mapping defined in Fig. 6: no concepts are added, modified, or removed. Table 6 shows a sample of the XML instantiations for the Servinsurance case, in DEMO (ex-ante model transformation) and ArchiMate (ex-post model transformation). ATL can produce an ArchiMate instantiation in XML, given: (1) the DEMO and ArchiMate meta models, defined in an ECORE syntax (ECORE is a meta model from Eclipse to describe models in terms of classes, attributes, data types, etcetera); (2) a meta model mapping, and (3) an instantiation of the DEMO meta model, defined in XML.

As stated in Sect. 4.2.2, ATL relies on transformation rules to create model transformations. For our chain, an example rule is:

```
rule Subject2BusinessActor{from s : DEMOMM!Subject to
t : ArchiMM!BusinessActor (name <- s.name)}
```

Note that currently, the tool implementation is a proof-of-concept, showing that the mapping that we presented in this paper can indeed be implemented into software. Thus, we showcase the *possibility* for creating a formal tooling

[3] http://www.eclipse.org/atl/.

Table 6 Sample of the XML instantiations for the Servinsurance

DEMO instantitation	ArchiMate instantitation
`<Subject name="Insurance Broker">`	`<BusinessActor name="Insurance Broker">`
`<performs_role`	`<assigned_to`
`name="Insurance Broker Role"/>`	`name="Insurance Broker Role"/>`
`</Subject>`	`</BusinessActor>`
`<Subject name="Customer">`	`<BusinessActor name="Customer">`
`<performs_role name="Customer Role"/>`	`<assigned_to`
`</Subject>`	`name="Customer Role"/>`

chain, whereby models created in a DEMO software tool[4] can be exported to a format interpretable for an ArchiMate tool.[5] However, the actual implementation of such a chain is outside of the scope of this paper.

7 Discussion

7.1 Limitation: DEMO transaction patterns introduce bias towards social perspective

Our application of DEMO transaction patterns to transform an e^3*value* model into an ArchiMate model biases the modelling of business processes towards a social perspective.

While modelling business processes, one can take different perspectives on the organization at hand [36]. This includes the Language Action Perspective (LAP), wherein one focuses on the social interactions between actors in an organization. Yet, another perspective can be an information-processing perspective, wherein one perceives of an organization as a collection of information-processing units that, by using formal transformation rules, transform input into output.

Such different perspectives have an influence on what is modelled in a business process [36]. For one, in LAP one focuses on modelling social interactions, thus for example emphasizing that the customer in the Servinsurance case is an active participant in creating an insurance policy. Yet, when emphasizing the LAP perspective one naturally pays less attention to consideration that are relevant from other perspectives. For example, from an information processing perspective, one would focus on issues such as the business rules that an insurance broker and insurer follow while creating an insurance.

It seems important to explicitly point out DEMO's bias on the social perspective of business processes. To what extent this bias leads us to miss important issues, is for now a direction for further research.

7.2 Planned practical validation

While the computational assessment discussed in Sect. 6 provides a proof-of-concept of our formal model transformation, it does not validate the claimed *advantages* of our model transformation. To address this issue, we foresee a practical validation with a major IT service provider, who has already expressed interest in comparing notes. The validation that we plan here is twofold, and builds on the advantages of our model transformation defined in Sect. 3:

Validating our claim that the value modelling in e^3value and the modelling of business processes and IT infrastructure in ArchiMate complement each other. This we can do by simply discussing with the IT services provider both a value and enterprise architecture view on the same system, and recording the responses from this. Feedback from a presentation of our model transformation work to representatives of the aforementioned large IT service provider is already promising, but their positive initial feedback needs to be further confirmed.

Validating our claim that better models are produced by our chain of model transformations. We claim that by using DEMO as a front end for ArchiMate we produce better ArchiMate models than without.

We aim to validate this claim by comparing the quality of the produced models with and without using DEMO as a front end, using datasets of the large IT service provider as input. To assess model quality, we will rely on the well-established SEQUAL framework [24]. SEQUAL is a framework dedicated to assessing the quality of conceptual models on a set of specific criteria, the most relevant for us being (1) *learning*, which expresses the delta in new, relevant, knowledge gained by stakeholders from the models, and (2) *the syntactic correctness of the models*. This we deem important because DEMO transaction patterns produce fine grained

[4] For example, while still in a test-phase, the browser-based, open-access, DEMO modelling environment available on 'modelworld' (http://www.modelworld.nl/, last accessed on April 17, 2012) allows for creating DEMO models.

[5] For example,the Open Source tool Archi (http://archi.cetis.ac.uk/) allows for creating ArchiMate-models, and importing/exporting these to XML-based formats.

facts, that do not always translate well to more coarse-grained ArchiMate process events.

In addition, we would also like to validate the traceability between *e*^{3}*value*, DEMO and ArchiMate models. This means that we validate—for different datasets from the IT services provider—the consistency of ArchiMate, DEMO and *e*^{3}*value* models created from our model transformation. Unfortunately, however, as opposed to evaluating the quality of models, the literature on evaluating the quality of model transformations themselves seems immature.[6] On the one hand, research on assessing the quality of a model transformation by means of a SEQUAL-like list of criteria is limited. We could find [34,35], both of which seem immature in the sense that (1) metrics for testing are always domain-specific, lacking generalization. For example, [34,35] find metrics by analyzing code for the ATL model transformation environment only. They lack generalization to a different context, such as alternative supporting software tools for model transformation. (2) the metrics are intuitive, lacking practical validation [35, p. 14]. In addition, there exists literature on the automated testing of model transformation, by means of a comparison of a test set of transformed models to an 'oracle' model transformation (see e.g. [21]). Here, 'oracle' refers to transformations that have been manually checked and approved by domain experts. Potentially, this work on model transformation testing could provide us with a measure of traceability for our work. However, also model transformation testing is in an early stage, as pointed out in the recent work of [1, p. 8]: "We need to develop techniques for the precise definition of requirements for model transformations. Currently, the requirements tend to be very informal and cannot be processed by tools to automatically generate the expected result." Thus, even if a good set of criteria for model transformation would exist (which, as discussed, also requires further research), using them in a (semi-)automated process to assess the traceability across multiple models would still be challenging.

8 Related work

The *e*^{3}*alignment* approach provides tools for actually creating business-ICT alignment. It does so by ensuring that conceptual models depicting strategic, value, process and ICT perspectives, respectively, on the value web at hand are consistent with one another [31]. However, this approach works only on a syntactic level. For instance, if the concept of an actor in *e*^{3}*value* and the concept of a swim lane in an UML activity diagram actually means the same is not a consideration. Derzsi et al. enable profitability calculations of an ICT-infrastructure by providing a meta model that links an IT infrastructure modelled in UML to *e*^{3}*value* [9]. This approach has more formality than e^3*alignment*, yet it focuses on a link between IT and value only. As a result, business processes are not a consideration while these are realistically cost carriers as well.

The Object Management Group (OMG) provides the standard Unified Modelling Language (UML) (version 2.0) [15]. UML mainly focuses on modelling control-flow and data perspectives [32]. It provides limited support for modelling organizational or value aspects of business processes.

Ontological mapping approaches address the semi-automated integration of system models [10]. System models are created in terms of a modelling language, which in itself is based on a meta model. Syntactic and semantic mapping between pairs of meta models has been facilitated by the application of existing approaches for ontology mapping [33]. Ontologies improve not only the semantics of a meta model but also provide a potential way in which these meta models can be bridged with each other to be integrated within a common context [16]. However, ontology mapping approaches focus on automating the discovery of a mapping, with less emphasis on the users that create the mapping [13]. Yet, in our research we require a precise mapping. Since our starting point are ontologies, such as *e*^{3}*value*, with relatively few concepts (compared to larger ones such as found in the medical domain), it seems better to perform mapping/transformation manually and as such, avoid an automatically generated approximation of a mapping.

9 Conclusion and future work

In this paper, we showed how to transform the value modelling technique *e*^{3}*value* to the enterprise architecture modelling technique ArchiMate, using as a bridge the DEMO method and modelling language. We introduced a formal mapping between the *e*^{3}*value*, DEMO and ArchiMate modelling techniques, and showed how this meta model mapping can be used to consistently transform between instantiations of these meta models. Also, we showed how the DEMO standard transaction pattern can aid in translating a high-level economic transaction into a business process. Finally, we provided a proof-of-concept software implementation of part of the proposed model transformation, to provide computational assessment of the proposed model transformation.

For future research, we intend to practically validate our model transformation with an interested large IT service provider. In addition, we will use the proposed model as input for further exploring the pragmatics of model transformation. This means that we will further explore context-dependent factors that come into play with model transformation, such as the overall objectives of the transformation to be achieved, and the concerns and skills of the stakeholders involved. In

[6] This is based on a literature search on google scholar, ingentaconnect, and citeseer with the key words model transformation {quality, assessment, testing, evaluation}.

particular, we aim to use the specific experiment from this paper as input for a formal 'method bundling' approach that can assess, for a candidate set of models, the return of the modelling effort. By this we mean that we make a formal cost/benefit analysis to assess to what extent the transformation of multiple models is beneficial for the parties involved, building upon earlier value modelling work [6].

Acknowledgments This work has been partially sponsored by the *Fonds National de la Recherche Luxembourg* (http://www.fnr.lu), via the CORE and PEARL programmes.

References

1. Baudry, B., Ghosh, S., Fleurey, F., France, R., Le Traon, Y., Mottu, J.M.: Barriers to systematic model transformation testing. Commun. ACM **53**(6), 139–143 (2010)
2. Bos, L., et al.: Finding the service you need: human centered design of a digital interactive social chart in dementia care (dem-disc). Med. Care Compunet. **5**(137), 210 (2008)
3. van Buuren, R., Gordijn, J., Janssen, W.: Business case modelling for e-services. In: 18th Bled eConference eIntegration in Action. AIS (2005)
4. Cummins, J.D., Doherty, N.A.: The economics of insurance intermediaries. J. Risk Insur. **73**(3), 359–396 (2006)
5. Czarnecki, K., Helsen, S.: Feature-based survey of model transformation approaches. IBM Syst. J. **45**(3), 621–645 (2006)
6. de Kinderen, S.: Needs-driven service bundling in a multi-supplier setting–the computational e^3service approach. PhD thesis, VU University Amsterdam (2010)
7. de Kinderen, S., Gaaloul, K., Proper, H.A.: Integrating Value Modelling into ArchiMate. In: Third International Conference on Exploring Service Science. Springer, Geneva, 125–139 (2012)
8. de Kinderen, S., Gaaloul, K., Proper, H.A.: On transforming DEMO models to ArchiMate. In: Proceedings of the 2012 EMMSAD/Eurosymposium workshop, Gdansk, Poland, pp. 270–284. Springer, Berlin (2012)
9. Derzsi, Z., Gordijn, J., Kok, K.: Multi-perspective assessment of scalability of it-enabled networked constellations. In: Sprague, R.H. (ed.) Proceedings of the 41st Annual Hawaii International Conference on System Sciences, p. 492. IEEE CS (2008)
10. Devedzić, V.: Understanding ontological engineering. Commun. ACM **45**, 136–144 (2002)
11. Dietz, J.L.G.: The deep structure of business processes. Commun. ACM **49**(5), 58–64 (2006)
12. Dietz, J.L.G.: Enterprise ontology: theory and methodology. Springer, Berlin (2006)
13. Falconer, S.M., Noy, N.F., Storey, M.A.: Ontology mapping–a user survey. In: Proceedings of the Workshop on Ontology Matching (OM2007) at ISWC/ASWC2007, Busan, South Korea, pp. 113–125 (2007)
14. Gordijn, J., Akkermans, H.: Value based requirements engineering: exploring innovative e-commerce ideas. Req. Eng. J. **8**(2), 114–134 (2003)
15. Object Management Group. UML 2.0 Superstructure Specification. Available at http://www.omg.org/cgibin/doc?ptc/2004-10-02. Accessed on 22 august 2012 (2004)
16. Happel, H., Seedorf, S.: Applications of ontologies in software engineering. In: 2nd International Workshop on Semantic Web Enabled Software Engineering (SWESE 2006), held at the 5th International Semantic Web Conference (ISWC 2006) (2006)
17. Iacob, M.-E., Jonkers, H., Lankhorst, M.M., Proper, H.A.: ArchiMate 2.0 Specification. The Open Group (2012)
18. Jonkers, H., Band, I., Quartel, D.: The ArchiSurance Case Study. White paper, The Open Group, Spring (2012)
19. Jonkers, H., Lankhorst, M.M., van Buuren, R., Hoppenbrouwers, S.J.B.A., Bonsangue, M., Van der Torre, L.: Concepts for modeling enterprise architectures. Int. J. Cooperative Inf. Syst. **13**(3), 257–288 (2004)
20. Stichting DEMO kenniscentrum. DEMO: The KLM case. http://www.demo.nl/attachments/article/21/080610_Klantcase_KLM.pdf. Last accessed on 22 August (2012)
21. Kessentini, M., Sahraoui, H., Boukadoum, M.: Example-based model-transformation testing. Autom. Softw. Eng. **18**(2), 199–224 (2011)
22. Klein, M.: Combining and relating ontologies: an analysis of problems and solutions. In: Workshop on ontologies and information sharing, IJCAI, vol. 1, p. 4. CEUR-WS (2001)
23. Kort, C., Gordijn, J.: Modeling strategic partnerships using the e3value ontology—a field study in the banking industry. In: Rittgen, P. (ed.) Handbook of Ontologies for Business Interaction, chapter XVIII. IGI Global, Hershey (2007)
24. Krogstie, J., Sindre, G., Jørgensen, H.: Process models representing knowledge for action: a revised quality framework. Eur. J. Inf. Syst. **15**(1), 91–102 (2006)
25. Lankhorst, M. M.: Viewpoints Functionality and Examples. Telematica Institute (2004)
26. Lankhorst M. M., et al.: ArchiMate Language Primer. Telematica institute (2004)
27. Lankhorst, M.M., et al.: Enterprise architecture at work: modelling. In: Communication and Analysis. Springer, Berlin (2005)
28. Lankhorst, M.M., Proper, H.A., Jonkers, H.: The architecture of the ArchiMate language, pp. 367–380. Enterprise, Business-Process and Information Systems Modeling (2009)
29. Levendovszky, T., Karsai, G., Maroti, M., Ledeczi, A., Charaf, H.: Model reuse with metamodel-based transformations. Software Reuse: Methods, Techniques, and Tools, pp. 166–178 (2002)
30. Op't Land, M., Middeljans, K., Buller, V.: Enterprise Ontology based Application Portfolio Rationalization at Rijkswaterstaat. In: The 4th Dutch Championship ICT, Architecture (2007)
31. Pijpers, V., Gordijn, J., Akkermans, H.: e3alignment: exploring inter-organizational alignment in networked value constellations. Int. J. Comput. Sci. Appl. **6**(5), 59–88 (2009)
32. Russell, N., van der Aalst, Wil M.P., ter Hofstede, Arthur H.M., Wohed, P.: On the suitability of UML 2.0 activity diagrams for business process modelling. In: APCCM '06: Proceedings of the 3rd Asia-Pacific Conference on Conceptual Modelling, pp. 95–104. Australian Computer Society, Inc., Darlinghurst (2006)
33. Shvaiko, P., Euzenat, J.: Ten challenges for ontology matching, pp. 1164–1182. Springer, Berlin (2008)
34. Van Amstel, M.F.: The right tool for the right job: assessing model transformation quality. In: IEEE 34th Annual Computer Software and Applications Conference Workshops (COMPSACW), pp. 69–74. IEEE (2010)
35. Vignaga, A.: Metrics for measuring ATL model transformations. MaTE, Department of Computer Science, Universidad de Chile, Tech. Rep (2009)
36. Winograd, T.: A language/action perspective on the design of cooperative work. Human-Computer Interact. **3**(1), 3–30 (1987)
37. Zivkovic, S., Kühn, H., Karagiannis, D.: Facilitate modelling using method integration: an approach using mappings and integration rules. In: Proceedings of the fifteenth European conference on information systems, ECIS 2007, pp. 2038–2049. AIS, St. Gallen (2007)

Author Biographies

Sybren de Kinderen (1982) is a postdoctoral researcher, and project manager, at the public research centre Henri Tudor in Luxembourg. His research interests are mainly focused on the pragmatics of conceptual modeling techniques (how do language complement each other, intended versus de facto language use, et cetera). Before joining Tudor as a postdoctoral student, Sybren de Kinderen did his PhD on the topic of "service bundling". As indicated by the thesis' title "Needs-driven service bundling in a multi-supplier setting—the computational e3-service approach" this work was largely interdisciplinary, in that it combined work from marketing ("needs-driven service bundling") with work from computer science ("the computational e3-service approach").

Khaled Gaaloul is a Postdoctoral Researcher at the Public Research Centre Henri Tudor in Luxembourg. Prior to this, he has done his PhD in the French National Institute for Research in Computer Science and Control (LORIA-INRIA) in the year of 2010. During his doctoral project, he has been working as a Research Associate at SAP Research in Karlsruhe, Germany. His main research interests include Enterprise Engineering, Service Innovation, Business Process Management and Security.

Henderik A. Proper Erik to friends, is a senior research manager at the Public Research Centre—Henri Tudor in Luxembourg, where he leads the Enterprise Engineering team (http://www.ee-team.eu). He also holds a chair in Information Systems at the Radboud University Nijmegen. Erik has a mixed background, covering a variety of roles in both academia and industry. He has co-authored several journal papers, conference publications and books. His main research interests include enterprise architecture, enterprise engineering, enterprise modelling, systems theory, business/IT alignment and conceptual modelling. His professional passion is the further development of the field of enterprise engineering and enterprise architecture. He can be contacted at: e.proper@acm.org.