REA-XML: An unambiguous language for REA business models

Dieter Mayrhofer, Christian Huemer
Business Informatics Group
Vienna University of Technology
Vienna, Austria
{*mayrhofer, huemer*}*@big.tuwien.ac.at*

Birgit Hofreiter, Christian Sonnenberg
Institute of Information Systems
University of Liechtenstein
Vaduz, Liechtenstein
{*birgit.hofreiter, christian.sonnenberg*}*@uni.li*

***Abstract*—Conceptual languages to describe business models - not business process models - become increasingly important. One of the most expressive approaches towards business models is the Resource-Event-Agent (REA) ontology. REA has a substantial back-up in economic and accounting theory, but did lack a dedicated representation language for many years. Recently, we have introduced a domain specific modeling language including a REA meta model and a pertinent graphical notation. In this paper we build upon our previous work, but aim for a representation language that is able to serve as a serialization syntax also for other REA modeling approaches using UML class diagrams or OWL, etc. As a result we come up with REA-XML, an XML schema that allows a precise, tool-independent representation of REA models and that may also serve as an REA model exchange language between different tools.**

I. Introduction

A business model is defined by Paul Timmers as an architecture for the product, services and information flows, including a description of the various business actors and their roles. Furthermore, it describes the potential benefits for the various business actors and the source of revenues [1]. Linder and Cantrell [2] share a similar point of view and define a business model as a company's core logic in order to create value by explaining how a company acts on the market and earns money. Additionally, they specify that a business model consists of distinct components which include and represent the essential business logic building blocks. These building blocks range from revenue models and value propositions to organizational structures and arrangements for trading relationships.

These fundamental business model definitions lay the cornerstone for the development of business modeling ontologies. Today, we see three main ontologies to conceptualize business models: the Business Model Ontology (BMO) [3], the e3-value ontology [4], and the Resource-Event-Agent ontology (REA) [5]. BMO concentrates on the categorization of aspects relevant for the delivery of products and services, but has a limited focus on conceptualizing the elements of the business model. Both, e3-value and REA deliver a well defined conceptualization of the elements of a business model. REA covers more concepts than e3-value and, thus, is more powerful. However, e3-value comes with a well-defined meta-model, a graphical syntax, and corresponding tool support. These features have been missing for REA for a long time, and as consequence REA did not gain as much user acceptance as e3-value.

Following this assessment, we have started the development of a precise modeling language for REA. In a first step, we have identified the relevant elements of the REA ontology and their relationships in order to derive a Meta Object Facility (MOF) compliant meta model. Given this meta model, we developed a dedicated graphical notation for REA business models. In other words, we did come up with a REA domain specific modeling language called REA-DSL [6]. Furthermore, at TU Vienna we implemented REA-DSL tool support based on *Microsoft's Visual Studio 2010 Visualization and Modeling SDK*.

Currently, the support for the REA meta model and the REA-DSL is limited to our tool. However, there may be further implementations in the future based on the underlying REA meta model, e.g. at the University of Liechtenstein we have started the development on top of Oryx. In order to come up with a REA description language that is tool independent one may think of using XML Metadata Interchange (XMI) which is designed to describe any meta data whose meta model can be expressed in MOF. However, XMI showed some deficiencies in practice since there exist a lot of different, incompatible, and vendor-specific XMI dialects and subsets. Furthermore, an XMI-based representation language will require the full support of our meta model, and will exclude other approaches such as the one by Gailly and Poels [7] who base their conceptual representation on stereotyped UML class diagrams (note these UML models may be expressed in XMI as well, but will be incompatible with our REA meta model). Others may use ontology editors based on RDF, OWL [8], etc. to describe REA models, which are not compatible with XMI at all. Accordingly, it seems to be appropriate to provide a serialization of the REA abstract syntax based on an XML schema dedicated to the REA concepts. Of course, we could have used an automatic transformation of our meta model to an XML schema by the Visual Studio SDK, but this would be a hoax, since it creates a proprietary Microsoft schema including a lot of overhead not acceptable to others. Thus,

we developed a dedicated XML schema for REA models, called REA-XML, that allows a precise, tool-independent representation of REA models and that may also serve as an REA model exchange language between different tools. In this paper we present the design of REA-XML.

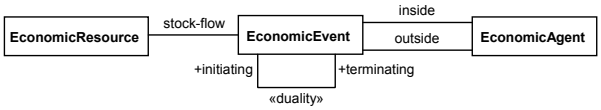
The remainder of this paper is structured as follows: in Section II we elaborate on related work that is dedicated to REA due to the specific focus of this paper. Since we build upon our previous work on the REA-DSL, this domain specific language is subject to Section III. In Section IV we present the REA-XML design by pointing to the equivalencies in the REA-DSL. A summary and remarks on future work conclude the paper in Section V.

II. Related Work on REA

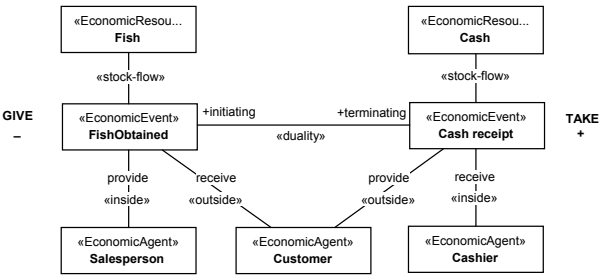
The objective of the REA ontology is the conceptualization of common economic phenomena of a firm independent of application-specific demands. REA accounting information systems focus on economic exchanges as the central unit of analysis. Instead of representing these exchanges with double-entry bookkeeping artifacts (e.g. debits, credits, accounts), REA proposes concepts and patterns to derive semantic models of economic exchanges and transformations. The underlying assumption of REA is that all business enterprises operate in the same manner [9] according to an entrepreneurial script: acquiring financial resources, engaging in a chain of economic exchanges with other parties, each time giving up an economic resource in return for another resource of greater value [9]. After executing this script, the business generates a justifiable profit after having paid interests and creditors. This entrepreneurial script essentially discloses the entrepreneurial rationale of a business. Hence, the REA ontology is not only used to facilitate the design of accounting information systems but in particular for business modeling.

The basic REA ontology is a stereotypical representation of an economic exchange as a core economic phenomenon [10]. This exchange is executed between parties inside and outside of a firm's boundaries and follows a particular object pattern (cf. [10], see also Figure 1(a) and 1(b)). In order to conceptualize this pattern, the REA ontology suggests three concepts that constitute an exchange: *resource*, *event*, and *agent*. Resources are things being exchanged between participating agents. In an exchange, an agent (inside agent) usually gives up control of a resource to an outside agent in order to gain control over another resource. Events occur in the course of executing economic activities. In REA basically two types of events are distinguished: *increment* and *decrement* events. Extensions of the REA ontology [11] also distinguish between *transfer* (exchanges with external actors) and *transformation* (concerns value creation within the firm) events.

Furthermore, the following economic primitives (relationship types) are specified by the REA: *duality*, *stock-flow*, and *participation*. A duality relationship connects decrement events with corresponding increment events and thus provides the rationale of individual economic activities. Stock-flow relationships connect economic resources with economic events (decrement or increment events). Depending on the connected event type, the following stock-flow relationship types are distinguished: *give* and *take* (transfer events), *use*, *consume*, and *produce* (transformation events). Participation relationships describe the involvement of an agent in an economic event. As such, the basic REA ontology not only conceptualizes economic exchanges but also relates to business process concepts. It captures *Who* is involved in an exchange (*economic agents*), *What* is being exchanged (*economic resources*), *When* (and under what conditions) do the components of an exchange occur (*economic events*), *Why* are the participants engaged in an exchange (*duality relationships*, *stock-flows*), and *How* do the exchanges materialize as economic activities or business processes (series of small events that move business process through to completion) (cf. [12]).



(a) REA Meta Model



(b) REA Object Constellation

Figure 1: Basic REA Ontology

The basic REA ontology with its economic primitives is illustrated in Figure 1(a) by means of a UML class diagram. Figure 1(b) shows an instantiation of the REA ontology, a so called object constellation. Furthermore, the ontology defines three axioms that restrict the use of the concepts and primitives for conceptualizing economic exchanges (cf. [11]):

- **Axiom 1**: At least one take event and one give event exist for each economic resource (guarantees modeling of economic activities as a sequence of exchanges).
- **Axiom 2**: All events effecting a resource decrement must be eventually paired in duality relationships with events effecting an increment and vice versa (ensures correct enumerations of exchanges).
- **Axiom 3**: Each exchange needs an instance of both the

”inside” and the ”outside” agents(ensures presence of exchanges between parties with competing economic interests).

Axioms one to three apply for transfer events. Axiom two also holds for transformation events.

Since its proposal in 1982, some extensions to the basic REA ontology have been proposed (e.g. [10]). The extended REA ontology envisions a vertical and horizontal layering of economic exchanges. With regard to the vertical layering, a hierarchy consisting of three levels is proposed: (1) the *value chain* specification level, (2) the *duality* specification level, and (3) the *task or workflow* level ([10]). The REA-DSL focuses on the upper two specification layers: the value chain and the duality.

The horizontal layering in REA enables the analysis of economic exchanges at different points in time on all three vertical layers described above. Therefore, the REA ontology considers an *accountability infrastructure* and a *policy infrastructure* [10]. The REA accountability infrastructure conceptualizes actual business events and captures ”what has occurred” or ”what is or has been”. The policy infrastructure conceptualizes what ”could be” or ”should be” within the context of a defined portfolio of a firm’s resources and capabilities [10]. The REA-DSL, which we will base the mapping on, realizes the concepts associated with the REA accountability infrastructure. Concepts associated with the policy infrastructure like commitments, agreements, and type images are not covered by the current REA-DSL but are subject to future work.

In the following section we show a REA-DSL example based on the REA ontology in order to demonstrate how REA constructs are used to specify an entrepreneurial script.

III. REA-DSL

Since the REA ontology introduced in the previous section is sometimes vague in the definition of the relationships between the core concepts and does not come with an easy to understand graphical notation, we developed a graphical domain specific language (DSL) for REA in [6]. This REA-DSL defines a precise graphical language for REA models. When moving to a serialization syntax that is independent of a graphical language, we may start from the underlying REA meta model defining the precise relationships between the core elements. Thus, this section presents the two most important views of the REA-DSL and the REA meta model. For educational purposes we decided to base our description on an illustrative example. The reader interested in all formal relationships is referred to our publication in [6]. The illustrative example has previously been used by Geerts and McCarthy [13] in explaining REA concepts and is based on a very simple, but real-world company:

Sy’s Fish is a distributor of seafood and provides his base of restaurant customers with over 50 types of fish which can be stored at all locations or stores. However, each store usually specializes in local favorites. Fish are purchased from local fishers, cleaned at the store, and then sold at restaurants to customers. Customers are allowed to buy on credit, and all pay on the last day of the month. Most employees are generalists and can perform many duties such as purchasing, cleaning, and delivering fish. They fill out time cards fortnightly upon which they may note the percentage of time devoted each day to buying, cleaning and selling fish. Non-generalist employees for the most part comprise of cashiers. Sy’s also possesses a fleet of trucks, used to bring fish from the docks and to deliver fish to the restaurants. Both the truck and the employees involved in each purchase and sale of fish are noted. All trucks are leased on yearly contracts, and lease payments are made monthly. Cash receipts and disbursements are made to/from one of the multiple checking accounts of the firm.

A. The Duality Model

As outlined in Section II, the concept of duality provides the rationale of individual economic activities. It links an increment in one set of resources with a corresponding decrement in another set of resources. These increments and decrements are triggered by economic events carried out by economic agents. The REA-DSL provides a dedicated meta model for specifying duality relationships, which is depicted in Figure 2. Note that the markers in this Figure - single letter in a black circle - become relevant in Section IV.

Instead of wading through the meta model, we decided to explain the duality concept by means of Figure 3 presenting a fish sale and compensating payment which is part of the *Sy’s Fish* example.

The *duality* concept applies to *transfers* (exchanges with external actors) and *transformations* (value creation inside

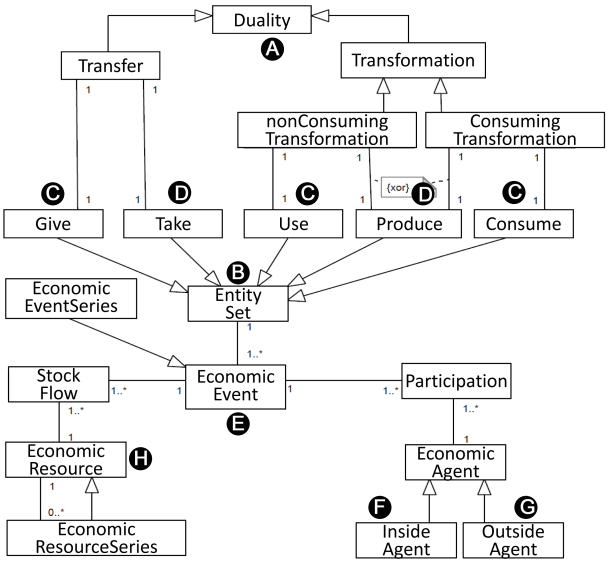


Figure 2: REA Duality Meta Model

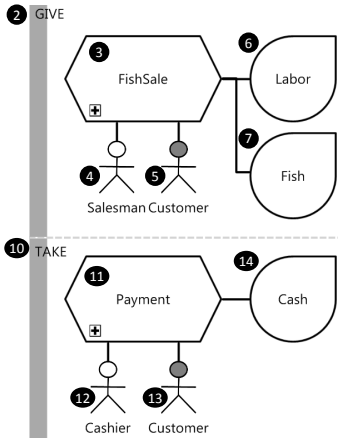


Figure 3: Sy's Fish Duality Model

the enterprise). No matter which kind of specialization it is, a *duality* always covers two distinct *entity sets*; one describing the increments in the resource sets and the other one the decrements in the resource sets. In our case of a *transfer* the decrement is called *give* and the increment is called *take*. *Give* and *take* are modeled as two separate partitions. The partition types are displayed in the upper left corner of the partitions.

An *entity set* covers at least one but up to multiple *economic events* (denoted by a hexagon). In our example the *economic events* are `FishSale` and `Payment`. An *economic event* is considered as a class of phenomena reflecting changes in scare means [14]. An *economic event* is specific to the *entity set* it belongs to. All *economic events* in the decrement *entity set* (*give*) decrement resources. They are counterbalanced by the *economic events* in the corresponding increment *entity set* (*take*) which increment resources.

The *economic resources* are denoted by drops (cf. `Labor`, `Fish`, and `Cash` in the example) and are connected with the *economic event* by so called *stock-flows*. An *economic event* may have one up to many *economic resources* connected. The same *economic resource* can be used in *multiple events* in various *duality models*.

An *economic event* involves *economic agents* (in our example `Salesman`, `Customer`, and `Cashier`). We distinguish between *outside agents* (denoted by a black stick-figure), i.e. trading partners outside the company, and *inside agents* (denoted by a white stick-figure) who are accountable inside the company. An *economic event* is associated to at least one, but up to many *economic agents*. The same *economic agent* can participate in *multiple events* in various *duality models*.

B. The Value Chain

The concept of duality described in the previous subsections serves as a building block for specifying value chains. According to Geerts and McCarty [13], the duality relationships introduced in the previous chapter are the glue that binds a company's economic events together into rational economic processes. Accordingly, a value chain is built by a well defined series of duality models. These duality models are connected by resources flows, meaning that a resource obtained in a preceding duality model is input for a subsequent duality model. Figure 4 presents the corresponding value chain meta model. A value chain is built by a number of business processes. Each business process is backed-up by a duality model serving as economic rational for this business process. The business processes are connected by economic resource flows.

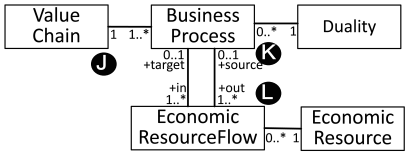


Figure 4: Value Chain Meta Model

Note, that the concept of a *business process* provides an interface to traditional business process modeling. In business modeling using REA a business process points to a duality model. In addtion the business process may point to a business process model specifiying a control flow of activties required to realize the value exchange. In this paper we do not further elaborate on this interface to busines process modeling.

Figure 5 depicts the value chain for the *Sy's Fish example* which is compliant to the meta model in Figure 4. This *value chain* is built by a number of *business processes* denoted by a rounded rectangle: `PayrollProcess`, `Buying`, `Truck-Acquisition`, `Transport`, `Cleaning`, and `Selling`. Each *business process* is backed-up by a *transfer* or a

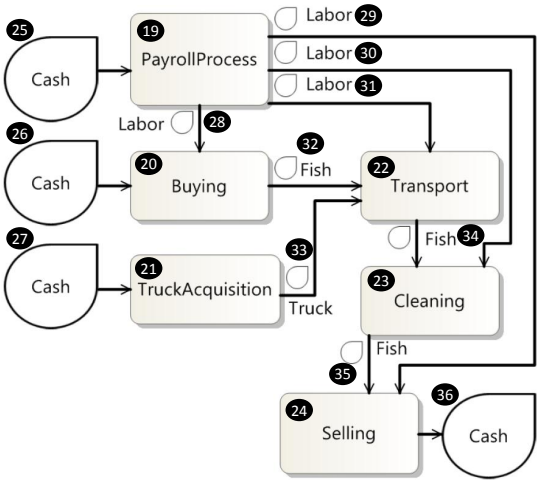


Figure 5: Sy's Fish Value Chain Model

transformation. It follows a *business process* points to an underlying *transfer/transformation* described by a *duality* model.

A *business process* is used only once in one distinctive *value chain*. A *business process* points to exactly one *duality* model. A *duality* model is usually the basis of one business process, but may be referred to by multiple *business processes*.

Economic resources (cf. `Labor`, `Fish`, `Truck` in the example) tie the *business processes* together (denoted by the small drops on the arrows). Thus, an *economic resource flow* - which points to exactly one *economic resource* - connects two *business processes*. An *economic resource flow* is a directed association that usually starts from a source *business process* and ends at a target *business process*. We might also have multiple resource flows going into or out of business processes (e.g. `Labor`, `Fish`, and `Truck` going into `Transport`). However, we also allow for *economic resource flows* that have either no source *business process* or no target *business process* (denoted by a big drop connected to the *business process*). This allows for a partial analysis, when one considers a certain *economic resource* as given or when an *economic resource* is considered as final output of the *value chain*. Typically, cash is often assigned to such *economic resource flows*. In our example we have the *resource flows* of `Cash` leading into `PayrollProcess`, `Buying`, and `TruckAcquisition` as well as *resource flow* of `Cash` starting from `Selling`.

A *business process* has at least one, but up to many outgoing *economic resource flows*. Similarly, a *business process* has at least one, but up to many ingoing *economic resource flows*. The *business process* `Selling` points to the *duality* model of Figure 3. In this *duality* model the *economic resources* `Labor` and `Fish` are in the *give* partition; and the *economic resource* `Cash` is included in the *take* partition. This is consistent with the *economic resource flows* to/from `Selling` in the *value chain* model of Figure 5. `Selling` uses `Fish` and `Labor` and creates `Cash`. It should be noted that the number of ingoing/outgoing transitions does not necessarily meet the number of *economic resources* in the *duality* model, since a *business process* may provide an *economic resource* or - more unlikely in practice - receive a *resource* from multiple *business processes*. For example, the *business process* `PayrollProcess` provides `Labor` four times to `Buying`, `Transport`, `Cleaning`, and `Selling`, but in the corresponding *duality* model, there is only one decrement resource.

IV. REA-XML

A. Mapping Overview

In the previous subsection we introduced the REA-DSL. At TU Vienna we implemented this REA-DSL on top of Microsoft Visual Studio, whereas the team in Liechtenstein based their implementation on Oryx. Even both implementations are based on the same DSL, the underlying tool specifics prevent an out-of-the-box REA model interchange. However, there exist other modeling approaches for REA models based on UML class diagrams [7] and OWL [8]. In this case, there is no common ground for a model interchange at all. Accordingly, there is a need for a common representation language that is able to serve as a serialization syntax for different REA modeling approaches. In this section we introduce REA-XML that allows a precise and tool-independent representation of REA models. REA-XML may be used as a native language to create REA models, but of course also serves as an exchange language for REA models between different tools.

Although REA-XML is designed to be tool independent, we started the design process of the REA-XML from our REA meta model (see Figures 2 and 4). This decision was due to the fact that the REA meta model is precise in the relationships between the REA core concepts overcoming the vagueness in the original REA ontology. Nevertheless, the resulting XML schema should not be influenced by any DSL-specifics and should provide a slim and concise tree structure format for REA models. Accordingly, we did not perform a pure UML-to-XML transformation, but aimed for a native XML design respecting all constraints present in the REA meta model. The design process of the REA-XML is further elaborated in the upcoming subsections.

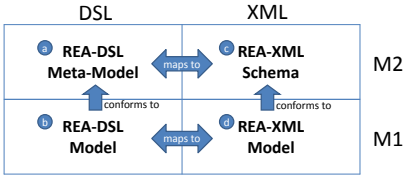


Figure 6: REA DSL-XML Mapping

In order to bind a tool specific implementation of REA to the REA-XML, a mapping to the REA-XML schema has to be implemented. In Figure 6 we present an overview of such a tool binding for our REA-DSL. On the meta model layer M2, a mapping from the REA-DSL meta model to the REA-XML schema, and vice versa, is specified. REA-DSL models on the model layer M1 are mapped to REA-XML models by executing the mapping specification of the M2 layer.

B. Mapping the Duality Model to XML

In this subsection we detail our design decisions when presenting the REA duality meta model in an equivalent REA-XML duality schema part. In order to help the reader in following our arguments, we have marked equivalent concepts in the REA duality meta model (Figure 2) and the duality XSD (Figure 7) with the same markers (white letter in a black circle). Furthermore, we decided to present

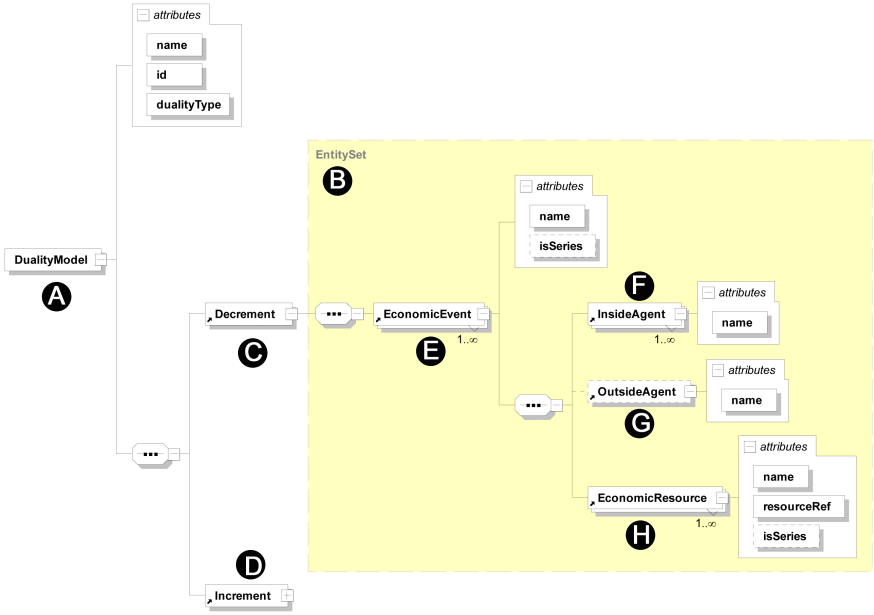


Figure 7: Duality XSD

the XSD in a graphical tree notation (used by XML Spy) rather then listing the XML code which is hard to ”parse” for humans. The reader interested in the XML code may access the XSD at http://umm-dev.org/rea-xml.

The concept of a `Duality model` (A) is mapped to the root element of the duality XSD part (note it is not the root of the overall XSD because this includes the value chain and the resources parts as well). In the XSD, the duality model has a `name` attribute and an `id` attribute for referencing purposes. In the meta model, both `duality` (A) and `entity set` (B) have subclasses. The relationship between `Duality` (A) and `Entity Set` (B) is modeled explicitly between these subclasses. However, when taking a detailed look one recognizes that a `Duality` (A) always covers two `Entity Sets` (B), one decrementing and one incrementing `Entity Set`. In case of a `Transfer` the decrementing one is `Give` (C) and the incrementing one is `Take` (D). In a `Non-Consuming Transformation` the decrement is `Use` (C) and `Produce` (D) is an increment. A `Consuming Transformation` evolves `Consume` (C) as a decrement and `Produce` (D) as an `Increment`. In the XSD, we decided that a `Duality` has two child elements: `Decrement` (C) and `Increment` (D) representing the two `entity sets`. Both `Decrement` (C) and `Increment` (D) are identically structured by inheriting from the complex type `Entity Set` (B). Furthermore, `Duality` (A) carries an attribute `duality type` which is an enumeration including the values `Transfer`, `nonCosumingTransformation` and `ConsumingTransformation`. Depending on the instance value of this attribute, one may derive whether the `Decrement` (C) is a Give, Use, or Consume and whether the `Increment` (D) is a Take or Produce.

In the meta model, an `Entity Set` (B) covers one or more `Economic Events` (E), and each `Economic Event` (E) resides in exactly one `Entity Set` (B). Accordingly, an `Entity Set` (B) in the XSD - no matter whether a `Decrement` (C) or an `Increment` (D) - includes one to many child elements representing `Economic Events` (E). The `Economic Event` (E) element has a `name` attribute and a boolean attribute `isSeries`. The latter is per default false, if true it indicates an `Economic Event Series`. In this paper we did not explain the concept of `Economic Event Series` in detail, thus the interested reader is referred to our paper on the REA-DSL [6].

By looking at the meta model, one recognizes that an `Economic Event` (E) has one to many `Participation` relations - each connecting exactly one `Economic Agent`, either an `Inside Agent` (F) or `Outside Agent` (G). From OCL constraints not depicted in the meta model, we know that an `Economic Event` (E) is connected to at least one but up to many `Inside Agents` (F) and, additionally, to no (in case of a transformation) or one (in the case of a transfer) `Outside Agent` (G). It follows that each `Economic Event` (E) element in the XSD has one to many `Inside Agent` (F) child elements and zero or one `Outside Agent` (G) child elements, both child elements carrying a `name` attribute.

Going back to the meta model, we see that an `Economic Event` (E) has one to many `Stockflow` relations - each connecting exactly one `Economic Resource` (H).

Thus, each `Economic Event` (E) element in the XSD additionally includes one to many `Economic Resource` (H) child elements. One should note that `Economic Resources` are re-usable objects in a REA model and, thus, are defined in a third main REA view, which is not further detailed in this paper due to its simplicity. Accordingly, the child element `Economic Resource` (H) of the element `Economic Event` (E) does not define a new resource, rather it includes a `resourceRef` attribute pointing to an economic resource in the resource view. For a better readability in the XML instance, the `Economic Resource` (H) child element still covers a `name` attribute. Furthermore, the boolean attribute `isSeries` is used to indicate an `Economic Resource Series` - this concept is similar to the `Economic Event Series` mentioned above.

Having explained the XML schema of the duality model in detail, we want to demonstrate a corresponding XML instance in Listing 1. Listing 1 is the XML equivalent of the DSL Duality Model `Selling` depicted in Figure 3. In order to better trace the correspondences, the DSL model (Figure 3) carries markers showing the line numbers of Listing 1.

Listing 1: Duality XML Model

```
1  <DualityModel name="Selling" dualityType="GiveTake" id="DU006">
2          <Decrement>
3                  <EconomicEvent name="FishSale">
4                          <InsideAgent name="Salesman"/>
5                          <OutsideAgent name="Customer"/>
6                          <EconomicResource name="Labor" resourceRef="RE001"/>
7                          <EconomicResource name="Fish" resourceRef="RE002"/>
8                  </EconomicEvent>
9          </Decrement>
10         <Increment>
11                 <EconomicEvent name="Payment">
12                         <InsideAgent name="Cashier"/>
13                         <OutsideAgent name="Customer"/>
14                         <EconomicResource name="Cash" resourceRef="RE003"/>
15                 </EconomicEvent>
16         </Increment>
17 </DualityModel>
```

C. Mapping the Value Chain Model to XML

Having presented the design decisions for the REA-XML duality schema part, we now move on to the value chain model part. Again, we use the same markers (single letter in a black circle) in the value chain meta model (Figure 4) and the tree-structure of the value chain XSD (Figure 8).

The concept of a `Value Chain` (J) is mapped to the root element of the value chain XSD part. In the XSD, the `Value Chain` (J) element carries a `name` attribute and an `id` attribute for referencing purposes. In the meta model, a `Value Chain` (J) includes one to many `Business Processes` (K). Accordingly, the `Value Chain` (J) element in the XSD includes one to many `Business Process` (K) child elements. Each `Business Process` (K) child element has a `name` attribute and an `id` attribute. In the meta model, a `Business Process` (K) is linked to exactly one underlying `Duality` model. To represent this referencing mechanism in the XSD, each `Business Process` (K) element includes a `dualityRef` attribute to carry a reference

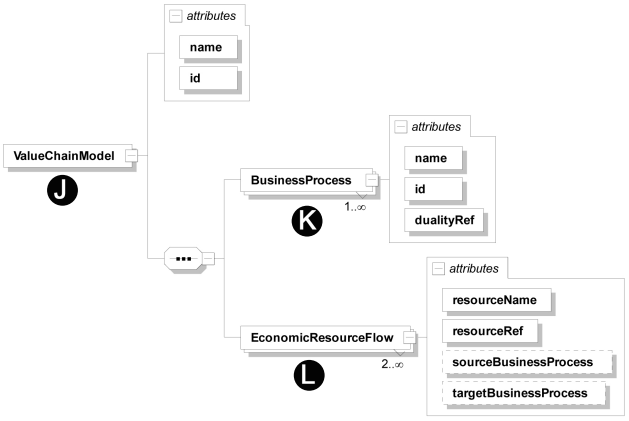


Figure 8: Value Chain XSD

(IDREF) matching the id of the corresponding `duality` model.

In the meta model, `Business Processes` (K) are related to `Economic Resource Flows` (L). A `Business Process` may be the source (out) and the target (in) of many `Economic Resource Flows` (L). Each `Economic Resource Flow` (L) starts usually from one `Business Process` (source), but may also have no source. Similarly, each `Economic Resource Flow` (L) leads usually to one `Business Process` (target), but may also have no target. In the XSD, we decided not to nest the `Economic Resource Flow` under the `Business Process`, because it usually connects two `Business Processes`. Instead, an `Economic Resource Flow` is a direct child element of the `Value Chain` (J) model. A `Value Chain` (J) model includes two to many `Economic Resource Flows` (L), at least one input to the `value chain` and one output of the `value chain`. An `Economic Resource Flow` (L) includes a `resourceRef` attribute to carry a reference (IDREF) matching the id of the corresponding economic resource. In addition to ensure better readability, the `resourceName` attribute carries the name of the resource. Furthermore, the `Economic Resource Flow` (L) element includes a `sourceBusinessProcess` and a `targetBusinessProcess` attribute, each carrying a reference (IDREF) matching the id of the source/target business process.

Listing 2 demonstrates an XML instance of the value chain model of the XML schema. This listing is the XML equivalent of the Sy's Fisch DSL value chain depicted in Figure 5, which also carries markers with the line numbers of the listing.

Listing 2: Value Chain XML Model

```
18 <ValueChainModel name="SysFish" id="VC001">
19         <BusinessProcess name="PayrollProcess" dualityRef="DU001" id="BP001"/>
20         <BusinessProcess name="Buying" dualityRef="DU002" id="BP002"/>
21         <BusinessProcess name="TruckAcquisition" dualityRef="DU003" id="BP003"/>
22         <BusinessProcess name="Transport" dualityRef="DU004" id="BP004"/>
```

```
23        <BusinessProcess name="Cleaning" dualityRef="DU005" id="BP005"/>
24        <BusinessProcess name="Selling" dualityRef="DU006" id="BP006"/>
25        <EconomicResourceFlow resourceName="Cash" resourceRef="RE003"
                targetBusinessProcess="BP001"/>
26        <EconomicResourceFlow resourceName="Cash" resourceRef="RE003"
                targetBusinessProcess="BP002"/>
27        <EconomicResourceFlow resourceName="Cash" resourceRef="RE003"
                targetBusinessProcess="BP003"/>
28        <EconomicResourceFlow resourceName="Labor" resourceRef="RE001"
                sourceBusinessProcess="BP001" targetBusinessProcess="BP002"/>
29        <EconomicResourceFlow resourceName="Labor" resourceRef="RE001"
                sourceBusinessProcess="BP001" targetBusinessProcess="BP004"/>
30        <EconomicResourceFlow resourceName="Labor" resourceRef="RE001"
                sourceBusinessProcess="BP001" targetBusinessProcess="BP005"/>
31        <EconomicResourceFlow resourceName="Labor" resourceRef="RE001"
                sourceBusinessProcess="BP001" targetBusinessProcess="BP006"/>
32        <EconomicResourceFlow resourceName="Fish" resourceRef="RE002"
                sourceBusinessProcess="BP002" targetBusinessProcess="BP004"/>
33        <EconomicResourceFlow resourceName="Truck" resourceRef="RE004"
                sourceBusinessProcess="BP003" targetBusinessProcess="BP004"/>
34        <EconomicResourceFlow resourceName="Fish" resourceRef="RE002"
                sourceBusinessProcess="BP004" targetBusinessProcess="BP005"/>
35        <EconomicResourceFlow resourceName="Fish" resourceRef="RE002"
                sourceBusinessProcess="BP005" targetBusinessProcess="BP006"/>
36        <EconomicResourceFlow resourceName="Cash" resourceRef="RE003"
                sourceBusinessProcess="BP006"/>
37  </ValueChainModel>
```

V. Summary and Future Work

In this paper we presented the REA-XML, an unambiguous and tool-independent language for REA models. Given its name REA-XML is based on XML and, thus, corresponds to an XML Schema as a textual, non-graphical notation for REA models. Using XML as a serialization syntax for REA models offers the potential (i) to use any XML editor for specifying REA models, and (ii) to use it as an exchange language between different REA tools. For the latter purpose, a transformation between the REA tool format and the REA-XML has to be implemented. Our team at TU Vienna has implemented a REA tool on top of Microsoft Visual Studio based on the REA-DSL. We have integrated the REA-XML as an import/export interface to the REA-DSL tool. This implementation facilitated the evaluation of the REA-XML presented in this paper by practical means. Firstly, the integration of REA-XML into the REA-DSL tool demonstrated the technical feasability of the approach and helped a lot in eliminating some flaws in the first drafts of the REA-XML. Furthermore, we wanted to test if the REA-XML properly supports the specifcation of existing REA models. Already for the evaluation of the REA-DSL tool, we had 32 REA models available and re-engineered them to represent them in the tool. This time, we exported these 32 REA models to REA-XML and imported the models into the tool again - this was all done without any semantic loss. Furthermore, we checked the REA-XML models manually for completness.

Currently, both the REA-DSL and the REA-XML are limited to the basic REA features. For future work, we will extend the REA-DSL to reflect more concepts of the REA ontology. Currently, we are working on an integration of the REA policy infrastructure [15] covering commitments, agreements and, furthermore, the typification of the operational concepts [10]. Additionally, we plan to attach attributes to the REA core concepts to enable monetary calculations and to derive a database design for enterprise information systems. Evidently, these extensions to the REA-DSL have to be reflected in the REA-XML in order to cope with the additonal REA concepts.

References

[1] P. Timmers, "Business models for electronic markets," *Electronic Markets*, vol. 8, 1998.

[2] J. Linder and S. Cantrell, "Changing business models: Surveying the landscape," *Accenture - Institute for Strategic Change*, May 2000.

[3] A. Osterwalder, A. Osterwalder, Y. Pigneur, Y. Pigneur, C. L. Tucci, and C. L. Tucci, "Clarifying Business Models: Origins, Present and Future of the Concept," *CAIS*, vol. 15, 2005.

[4] J. Gordijn and H. Akkermans, "E3-value: Designing and evaluating e-business models," *IEEE Intelligent Systems*, vol. 16, 2001.

[5] W. E. McCarthy, "The REA Accounting Model: A Generalized Framework for Accounting Systems in a Shared Data Environment," *The Accounting Review*, vol. 57, no. 3, 1982.

[6] C. Sonnenberg, C. Huemer, B. Hofreiter, D. Mayrhofer, and A. Braccini, "The REA-DSL: A Domain Specific Modeling Language for Business Models," in *23rd International Conference on Advanced Information Systems Engineering*, 2011.

[7] F. Gailly and G. Poels, "Ontology-driven business modelling: Improving the conceptual representation of the rea ontology," in *Conceptual Modeling - ER 2007*, 2007.

[8] ——, "Towards Ontology-Driven Information Systems: Redesign and Formalization of the REA Ontology," in *BIS*, 2007.

[9] G. L. Geerts and W. E. McCarthy, "An Accounting Object Infrastructure for Knowledge-Based Enterprise Models," *IEEE Intelligent Systems*, vol. 14, 1999.

[10] ——, "An ontological analysis of the economic primitives of the extended-REA enterprise information architecture," *International Journal of Accounting Information Systems*, vol. 3, 2002.

[11] ——, "The Ontological Foundations of REA Enterprise Systems," August 2000. [Online]. Available: http://www.msu.edu/user/mccarth4/Alabama.doc

[12] T. Motal and R. Schuster, "From e3-value to rea: Modeling multi-party ebusiness collaborations," in *Proc. of the 11th IEEE Conf. on Commerce and Enterprise Computing*, 2009.

[13] G. L. Geerts, , and W. McCarthy, "Modeling business enterprises as value-added process hierarchies with resource-event-agent object templates," in *In Business Object Design and Implementation*, 1997.

[14] S. C. Yu, *The Structure of Accounting Theory*. The University Presses of Florida, 1976.

[15] G. L. Geerts, , and W. McCarthy, "Policy-level specification in rea enterprise information systems," *Journal of Information Systems*, vol. 20, 2006.