The REA-DSL: A Domain Specific Modeling Language for Business Models

C. Sonnenberg[2], C. Huemer[1], B. Hofreiter[2], D. Mayrhofer[1], A. Braccini[3]

[1]TU Vienna, [2]University of Liechtenstein,[3]LUISS University
[1]{last}@big.tuwien.ac.at, [2]{first.last}@hochschule.li, [3]abraccini@luiss.it

Abstract. In the discipline of accounting, the resource-event-agent (REA) ontology is a well accepted conceptual accounting framework to analyze the economic phenomena within and across enterprises. Accordingly, it seems to be appropriate to use REA in the requirements elicitation to develop an information architecture of accounting and enterprise information systems. However, REA has received comparatively less attention in the field of business informatics and computer science. Some of the reasons may be that the REA ontology despite of its well grounded core concepts is (1) sometimes vague in the definition of the relationships between these core concepts, (2) misses a precise language to describe the models, and (3) does not come with an easy to understand graphical notation. Accordingly, we have started developing a domain specific modeling language specifically dedicated to REA models and corresponding tool support to overcome these limitations. In this paper we present our REA DSL which supports the basic set of REA concepts.

1 Introduction

Analyzing the economic phenomena on which companies base their business may serve as a good starting point in the requirements elicitation phase when developing enterprise information systems. Business models specify - amongst other things - the main actors, their relationships and the values exchanged between them (cf. [1]).

We see three main ontologies to conceptualize business models: the Business Model Ontology (BMO) [2], the e3-value ontology [3], and the Resource-Event-Agent ontology (REA) [4]. BMO is easy to use by the domain expert because it focuses mainly on the categorization of aspects relevant for the delivery of products and services to fulfill customers' requests. It helps the domain expert to ask herself the right questions when developing a business model, but has a limited focus on conceptualizing the elements of the business model. In contrary, e3-value defines a conceptual model to describe the exchanges of value among actors in a network. e3-value comes with a graphical syntax that is easy understood by the domain expert. Furthermore, it allows the domain expert to perform financial assessment of the value exchanges.

The Resource-Event-Agent (REA) ontology has its roots in the accounting discipline and was originally developed as a reference framework to conceptualize

economic phenomena in an enterprise. In its proposal in 1982, McCarthy already had the vision to facilitate the design of data structures of accounting information systems by means of REA [4]. Since this time the REA model has been further extended and evolved into a domain ontology [5]. All REA concepts are based on well established concepts of the literature in economic theory - which is certainly one of the strengths of REA. However, REA has no dedicated representation format and, consequently, no graphical syntax. Thus, users may struggle when describing the REA models leading to the impression that REA is a rather heavy-weight approach. McCarthy feels (as he expressed during our joint work in standardization activities) that a dedicated graphical syntax - such as it exists for e3-value - may help in overcoming this problem and may lead to a much more significant adoption of REA. Accordingly, we have started the endeavor of developing a domain specific modeling language for REA.

A domain-specific language (DSL) is a small, usually declarative language. It offers expressive power through appropriate notations and abstractions focused on – and usually restricted to – a particular problem domain [6]. Besides textual DSLs, we see an increasing interest in domain-specific modeling languages [7, 8]. According to [6] the benefits of a DSL approach are manifold: They allow solutions to be expressed at the level of abstraction of the problem domain. As a matter of fact, domain experts themselves can understand, validate and often modify DSL programs/models. The DSL programs/models are concise and self documenting to a large extent. They enhance productivity, reliability and maintainability. DSLs allow for validation and optimization at the domain level.

When developing our REA-DSL we followed methodological steps that have been suggested by Strembeck and Zdun for the systematic development of domain specific languages [9]. Amongst other variants, they describe the development process for extracting a DSL from an existing system, which is appropriate for our needs, because we extract the DSL from the existing REA ontology. Accordingly, we started with (1) the identification of elements in the REA ontology. Next, we underwent a number of revision cycles of (2) deriving the abstract syntax of the REA model including the core language model and the language model constraints and (3) defining the DSL behavior, i.e. determining how the language elements of the DSL interact to produce the intended behavior. Once we had reached a stable state, we defined the DSL concrete syntax (4). Finally, we implemented a modeling tool support for the DSL (5), but we skipped the last step described in [9], i.e. integrating the DSL into a software platform, since REA stays at the platform independent level.

The remainder of this paper is structured as follows: In Section 2 we give a basic introduction into the REA ontology including an illustrative example. The core of the paper in Section 3 presents our DSL for REA models. We elaborate on the meta model of our REA DSL for describing duality models and value chains. Furthermore, we use the same example as in Section 2 to illustrate our DSL. Section 4 reports on our evaluation by means of a REA DSL tool. A summary and remarks on future work conclude the paper in Section 5.

2 Resource - Event - Agent (REA)

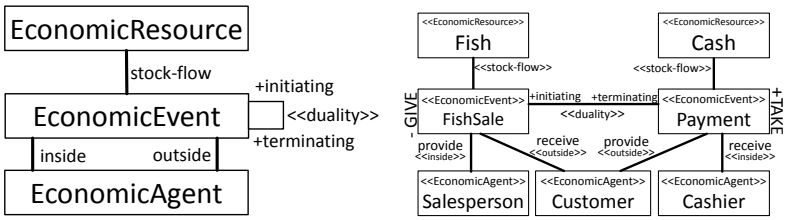
2.1 The REA Ontology

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 [10] 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 [10]. 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 [5]. This exchange is executed between parties inside and outside of a firm's boundaries and follows a particular object pattern (cf. [5], 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

Fig. 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. [5]). 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 ([5]). In this paper we focus on the upper two specification layers: the value chain and the duality . Thereby, we base our work on the diagram style used by Geerts and McCarty in [13], which has never been formalized.

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* [5]. 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 [5]. In this paper we focus on the concepts associated with the REA accountability infrastructure. Concepts associated with the policy infrastructure like commitments, agreements, and type images are not covered here but are subject to future work.

In the following section the REA ontology is applied to a simplified example in order to demonstrate how the REA constructs can be used to specify an entrepreneurial script.

2.2 REA Ontology example

The simplified example used to apply the REA ontology is taken from [13]. The example is based on an actual company and is called *Sy's Fish* and is introduced below:

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.

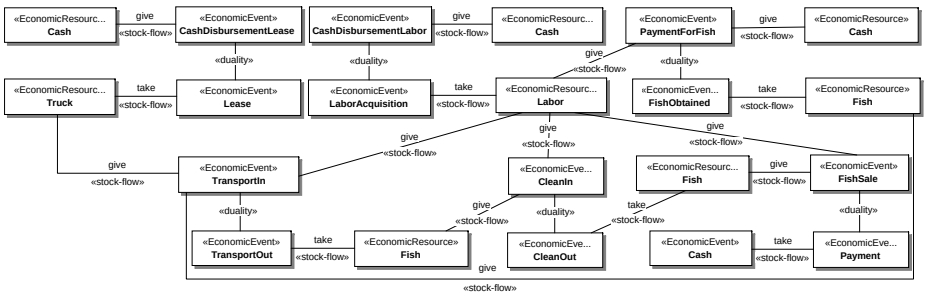


Fig. 2. REA Sy's Fish Example

The value chain of Sy's Fish (i.e. the entrepreneurial script) comprises the processes payroll, truck acquisition and store processing. The last process is further decomposed into the processes buying, cleaning, and selling. The process

level specification is provided in 2 as a UML class diagram. It depicts the object constellations for the entire value chain. Due to space limitations, economic agents have been omitted from the process level specification. The process level specification with its sequence of duality relationships determines the rationale of how Sy's makes money. Initial outlays (cash disbursements) are followed by subsequent cash flows. These cash flows can be used to pay creditors and interests or to finance further economic activities.

2.3 Limitations of the REA Ontology

A first limitation of REA is given with regard to the clarity of ontological statements. Although the REA model has evolved into a domain ontology, REA leaves space for diverging interpretations of the relationships between core concepts. In particular, the multiplicities for individual relationships are not clearly defined. It remains unclear, whether a resource increment or decrement is caused by one event or by multiple events. Moreover, it is not clear if an event can be related to multiple stock-flows and agents. Furthermore, there is no axiom restricting event-resource (stock-flow) relationships. In its original specification, REA would allow for simultaneously connecting both increment and decrement stock-flows with a single event. However, stock-flows have to be compatible with their associated events. A decrement event should only be related with decrement stock-flows and vice versa. Connecting an increment stock-flow with a decrement event would result in semantic inconsistencies.

These ambiguities and potential semantic inconsistencies may be due to a lack of a dedicated specification language. In the current state no means have been proposed to specify precisely how to relate the REA concepts and how to enforce compliance of REA models with the axioms stated above. The compliance of REA models can only be checked manually. Thus, domain experts are responsible for including the REA pattern into particular enterprise information architectures. There is no conceptual facility that enables the development of interoperable REA models (a modeling language would help here).

A third limitation is the complexity of REA models which increases progressively even with very few processes to be modeled. Each process generally includes eight entities which have to be modeled (two events, two resources, two times an inside and an outside agent) [10]. A dedicated REA modeling notation would help to overcome the complexity. However, proposing a modeling notation necessitates resolving the inconsistencies and incompleteness of the ontological statements. This is subject to the following sections.

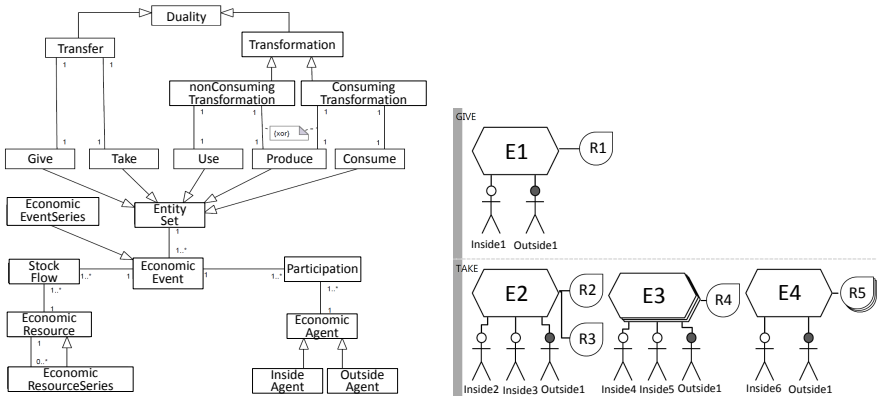
3 The REA Domain Specific Language

Given these limitations we started to develop a graphical domain specific language (DSL) for REA. We based the development of the REA DSL on The Object Management Group's(OMG) metamodeling architecture called Meta-Object Facility (MOF) [14]. MOF comes with a meta-meta model (M3 layer)

that allows us to define the structure, i.e. the abstract syntax, of the REA DSL as a meta-model (M2 layer). The resulting REA DSL meta-model comprises three interlinked views, of which we describe in detail the *duality* and the *value chain* perspective. Due to space limitations, we do not elaborate in detail on the third view on economic resources. However, it is important to note that economic resources - scarce objects having utility and under control of an enterprise [15] - may form a generalization hierarchy.

3.1 The Duality Model

The duality relationship is a core concept in REA. It links an increment in the resource set with a corresponding decrement; where increments and decrements must be members of two different event entity sets. In the REA ontology, the duality relationship is characterized by the unary relationship assigned to the concept of an economic event (see Figure 3(a)).



(a) Meta Model (b) Example

Fig. 3. Duality

In the REA DSL, *duality* becomes a core model element that serves as a building block for further purposes. The duality meta model is depicted in Figure 3(a). Figure 3(b) presents a (rather abstract) example model - which should help understanding the meta model concepts and to introduce the concrete syntax and the corresponding stencils.

The *duality* concept applies to *transfers* (exchanges with external actors) and *transformations* (value creation inside the enterprise). In the case of *transformations* we distinguish between *resource-consuming* and *non-resource consuming transformation*. As a consequence, the meta model defines *consuming transformation* and *non-consuming transformation* as special kinds of *transformation* as well as *transfer* and *transformation* as specializations of *duality*.

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 the case of a *transfer* the decrement is called *give* and the increment is called *take*. The decrement of a *non-consuming transformation* is denoted as *use* and the one of a consuming transformation is denoted as *consume*. In both kinds of *transformation* the increment is called *produce*.

Figure 3(b) shows an abstract example of a *duality* model to illustrate the concepts and their stencils. An entity set is modeled as a partition. Accordingly, a duality model includes two partitions. The *duality* shown in the example is a *transfer*. It follows that one *entity set* is a *give* and the other one is a *take*. The kind of *entity set* is denoted in the upper left corner of the partition.

According to the meta model, an *entity set* covers at least one but up to multiple *economic events*. An *economic event* is considered as a class of phenomena reflecting changes in scare means [16]. An *economic event* is specific to the *entity set* it belongs to. Following the principles of *duality*, all *economic events* in the decrement *entity set* (*give*, *use*, *consume*) are counterbalanced by the *economic events* in the corresponding increment *entity set* (*take*, *produce*) of the same *duality* model.

An *economic event* is usually executed at a certain point in time. However, in certain cases the increment/decrement is realized by a series of economic events each of the same nature. For example, consider that the payment for goods is split into a number of partial payments. For this purpose we use the concept of an *economic event series*, which is defined as a specialization of the *economic event* in the meta model. Consequently, the concept of an *economic event series* may substitute an *economic event*. This means, instead of modeling each *economic event* of each partial payment, one may model a single *economic event series* of partial payments.

An *economic event* in a *give/use/consume entity set* decrements resource(s). Similarly, an *economic event* in a *take/produce entity set* increments resource(s). This relationship between an *economic event* and an *economic resource* is described by the concept of a *stock-flow*. A *stock-flow* models an association between exactly one *economic event* and exactly one *economic resource*. An *economic event* will affect most of the time one *economic resource* only, but it may affect multiple ones. Thus, an *economic event* may have one up to many *stock-flows* connected. An *economic resource* usually is affected by many different *economic events* (in different *entity sets* of different *duality* models). At a minimum an *economic resource* is affected by one *economic event* - otherwise it would not be worth considering the *economic resource* at all. Consequently, an *economic resource* is connected to one up to many *stock-flows*.

A similar concept to *economic event series* is defined for *economic resources*: the *economic resource series*. An *economic resource series* is used if an *economic event* affects a number of *economic resources* of the same kind. For example, on a high level of abstraction one may define raw material as an *economic resource* and an *economic event* may affect a number of *economic resources*. It is important

to note that an economic resource series may substitute an economic resource in a stock-flow, but the economic resource series must not be used in economic resource generalization hierarchies. This is prohibited by a corresponding OCL constraint to the meta model. An *economic resource series* must always be based on exactly one existing *economic resource*. For an *economic resource* one may define zero to many *economic resource series* (used in different *stock-flows*).

An *economic event* involves *economic agents*. We distinguish between *outside agents*, i.e. trading partners outside the company, and *inside agents* who are accountable inside the company. The involvement of *economic agents* in *economic events* is denoted by the concept of *participation*. A *participation* is an association that connects exactly one *economic event* with one *economic agent*. An economic event is associated to at least one, but up to many economic agents. Hence, an economic event has one to many participation associations. An economic agent participates in at least one, but up to many economic events (in the same, but also in different entity sets of the same or different duality models). Thus, an economic agent has one to many participations connected.

In addition, there are further constraints assigned to the meta model to handle specifics of *transfers*. In case of a *transfer*, each *economic event* must be assigned to exactly one *outside agent* and, in addition, to at least one *inside agent*. All *economic events* of the same *transfer* (both in the *give* and the *take entity set*) must involve one and the same *outside agent*.

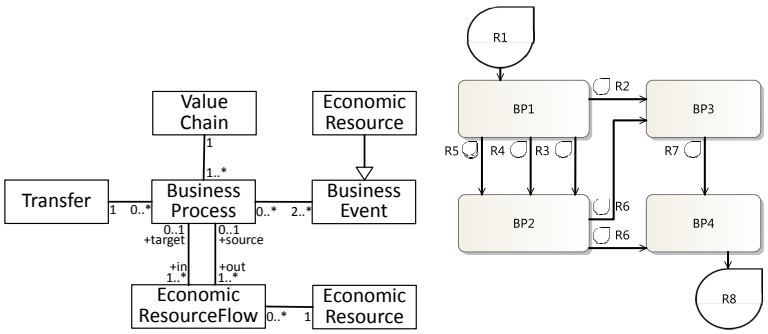
The relationships among *economic events*, *economic resources*, and *economic agents* within an *entity set* is also illustrated in the abstract example of Figure 3(b). *Economic events* are denoted by an hexagon. In the *give* partition, there is only one *economic event* E1. This *economic event* leads to a decrement of the only associated *economic resource* R1, notated by a drop. Since the *duality* model is a *transfer*, exactly one *outside agent* Outside 1 and an *inside agent* Inside 1 is involved. The symbol for *economic agents* is a stickman - *outside agents* have a black head, whereas *inside agents* have a white head.

The *take* partition includes three *economic events* E2, E3, and E4; one of which (E3) is an *economic event series*. The symbol of an *economic event series* is a pack of hexagons. All three *economic events* together compensate the single *economic event* E1 of the *give* partition. E2 is associated with two *economic resources* R2 and R3 that are incremented, whereas the series E3 increments only the *economic resource* R4. E4 leads to an increase of the *economic resource series* R5. An *economic resource series* is depicted by a pack of drops. Given the example of a *transfer*, all three events in the *take* partition must be associated with the same *outside agent* as the *economic event* of the *give* partition: Outside 1. In addition, E2 and E3 involve two *inside agents* and E4 only one.

3.2 The Value Chain

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, while "stock-flow" relationships weave these processes together into an enterprise value chain. In the latter case, they

do not refer to the stock-flows within a single duality model. Rather, they mean that an increment in a resource as a result of one duality model will serve as a chance to decrement this resource in a subsequent duality model. In other words, a value chain is built by a well defined series of duality models. The transitions between the duality models reflect the resource dependencies between the duality models.



(a) Meta Model (b) Example

Fig. 4. Value Chain

These general ideas are reflected in the value chain meta model of Figure 4(a). A *value chain* is built by a number of *business processes*. A *business process* is defined as a *transfer* or a *transformation*, and in addition the tasks needed to execute the *transfer/transformation*. It follows a *business process* points to an underlying *transfer/transformation* described by a *duality* model. Furthermore, a *business process* results in a number of *business events*, two of which are the *economic events* of the underlying *duality* model.

A *value chain* includes one to *many business processes*. 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*. A *business process* involves at least two *business events*, these are the two *economic events* (which are considered as special kinds of *business events*). Usually, there will be more *business events* associated to a *business process* as a result of the tasks needed to execute the *transfer/transformation*.

Economic resources tie the *business processes* together. 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*. However, we also allow for *economic resource flows* that have either no source *business process* or no target *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*.

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*. In order to deliver a consistent *value chain* two important constraints on the business processes have to be considered: For each *economic resource* in the *give / use / consume entity set* of the underlying *duality* model, at least one corresponding incoming *economic resource flow* pointing to the same *economic resource* must exist. In analogy, for each *economic resource* in the *take / produce entity set* of the underlying *duality* model, at least one corresponding outgoing *economic resource flow* pointing to the same *economic resource* must exist. However, one may consider the substitutability concept of more general and more specific economic resources as defined in a resource generalization hierarchy. In other words, a duality model expecting an economic resource in *give / use / consume entity set* may also accept a more specific *economic resource* in the incoming *economic resource flow*.

In Figure 4(b) we depict an abstract example of a *value chain* to illustrate its concepts and their stencils. Our *value chain* example includes four *business processes* BP1, BP2, BP3, and BP4. The symbol for a *business process* is a rounded rectangle. *Economic resource flows* are depicted by an arrow with the *economic resource* assigned to this arrow. For example, the *economic resource* R2 flows from BP1 to BP3. However, for esthetic reasons we provide an alternative (but still similar) notation for *economic resource flows* that do not start from or do not end in a *business process*. In this case, the arrow may start from the *economic resource* or may lead to the *economic resource*, instead of assigning the *economic resource* to the arrow. Examples are the resource flows of R1 leading into BP1 and of R8 starting from BP4.

BP1 points to the *duality* model of Figure 3(b). In this *duality* model the *economic resource* R1 sits in the *give* partition; and the *economic resources* R2, R3, R4, and R5 - where the latter is a *economic resources series* - are included in the *take* partition. This is consistent with the *economic resource flows* to/from BP1 in the *value chain* model of Figure 4(b). BP1 receives R1 and delivers R2, R3, R4, and R5. 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, BP2 provides R6 to both BP3 and BP4.

Let us assume that the *duality* model on which BP2 is based requires the *economic resources* R3, R5, and R7. In this case one might think of an inconsistency since BP2 receives R3 and R5, but R4 instead of R7. However, if R4 is defined as a specialization of R7 in a resource generalization hierarchy, the model is consistent since in this case R4 may substitute R7 in the *duality* model of BP2.

3.3 REA DSL Example

Having introduced the meta models and rather abstract examples, we now demonstrate our REA DSL by means of a simple, but more realistic example. For this purpose we use again the Sy's Fish example [13] of subsection 2.2. The resulting Sy's Fish value chain and duality models are depicted in Figure 5.

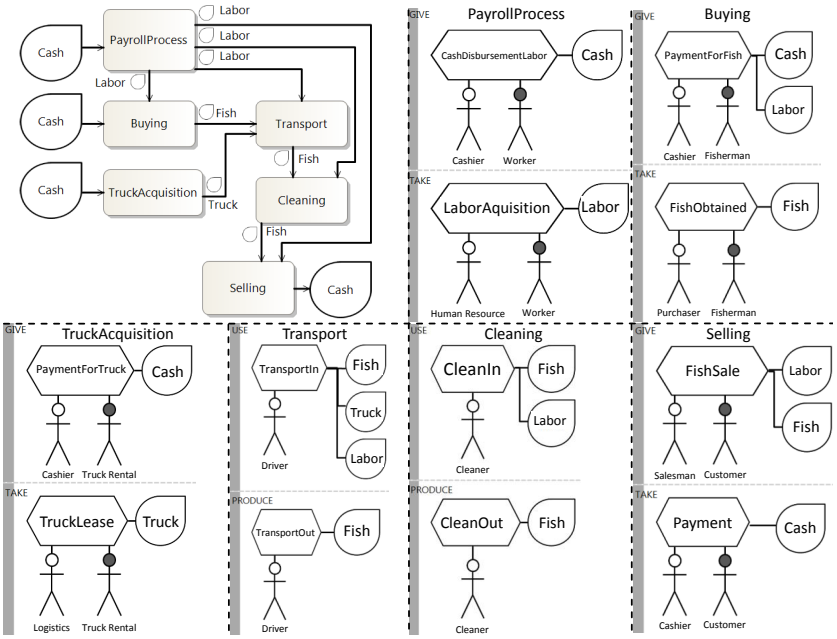


Fig. 5. Sy's Fish Value Chain and Dualities

The value chain begins with the `payroll process` which receives some `cash` to realize a transfer. By the duality relationship of the economic events `cash disbursement` and `labor acquisition`, the economic resource `cash` is decremented and *labor* is incremented. The resulting labor is then input for the business processes `buying`, `transport`, `cleaning`, and `selling`.

`Buying` transfers `labor` and additionally `cash` to `fish`, by the dual economic events `payment for fish` and `fish obtained`. `Truck acquisition` is another transfer that turns `cash` into a `truck` that is leased by the duality of `payment for truck` and `truck lease`.

`Transport` uses all acquired resource - `labor`, `fish`, and `truck` - to deliver the fish to the company. The `transport` is done by the company itself, so it does not involve an outside actor. Accordingly, it is a transformation process that uses, but does not consume its economic resources in the economic event `transport in`. The dual economic event `transport out` results again in the `fish`, this time

at the right place. `Cleaning` is another non-consuming transformation process that receives the `fish` and turns it into a cleaned `fish`.

The final transfer in the value chain is `selling`. `Labor` and `fish` are given in the `fish sale` in order to take `cash` as result of the `payment`.

4 REA DSL Tool Support

Having developed our DSL approach towards REA modeling in theory, we wanted to evaluate our approach by practical means. Thereby, we focused on three major steps. Firstly, we evaluated the technical feasibility of the DSL by an implementation based on Microsoft DSL tool kits. Thereby, we were able to eliminate technical flaws in the meta model. Secondly, we wanted to test our tool if it properly supports existing REA models. For this purpose we had 32 REA models available and successfully completed the reengineering task to represent these models within our REA DSL. During our reengineering activities the strengths of our rather strict meta model became evident - we were able to recognize flaws in the existing REA models which have not been recognized before due to the complexity in the ontology representation and missing tool support. Thirdly, we approached the originator of REA - William McCarthy - to seek his advise and to report inconsistencies in existing REA models. McCarthy sanctioned our DSL approach and provided valuable comments which we integrated into our approach. However, so far we have not yet done any usability studies, nor have we used the tool set in real world case studies. We will conduct these kinds of evaluations once we extend our approach to cover the REA policy infrastructure.

We implemented the graphical REA-DSL tool based on *Microsoft's Visual Studio 2010 Visualization & Modeling SDK (V&M SDK)*. Accordingly we used the V&M SDK to create the meta models explained before in Section 3. Additional custom code enables to set further constraints, necessary for the validation of the REA model. In a second step the designer - see Figure 6 - is created to support the REA modeling.

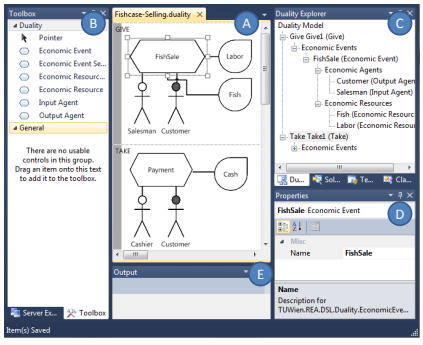


Fig. 6. REA-designer

The designer is separated into five major areas: The *modeling canvas* (A) the *toolbox* (B), the *solution explorer* (C), the *property window* (D) and the *validation window* (E). By dragging the modeling elements from the *toolbox* on the *modeling canvas*, a REA model can be assembled in a graphical representation. The *solution explorer* provides a tree based overview of the elements of the currently displayed model as well as a file and directory structure to hold different model instances. Properties of the selected model element can be changed in the *property window*. The *validation window* informs the user of any errors/warnings.

5 Summary and Future Work

In this paper we proposed a domain specific modeling language to support the conceptual modeling of economic events based on the REA ontology. Conceptual models based on our modeling language – the REA-DSL – aim at facilitating the requirements elicitation process during the design of accounting and enterprise information systems. The original REA ontology has a long history in accounting and is based on well grounded accounting concepts. However, REA leaves space for diverging interpretations of the relationships between core concepts. This has also been criticized by Gailly and Poels who have proposed a new conceptual representation of REA guided by proven ontology engineering principles [17]. They come up with a presentation based on UML class diagrams, which we feel results in a complex visual representation that is hardly understood by domain experts (cf. Figure 2). Similar to Gailly and Poels, we developed a representation format that does not leave space for divergent interpretations. In our case, the relationships between the core concepts are precisely defined by using OMG's Meta Object Facility (MOF) leading to a dedicated REA domain specific modeling language. The MOF-based approach enables the development of a graphical syntax that is dedicated to the needs of business modeling. Furthermore, our REA-DSL comes with a graphical syntax covering a set of stencils that facilitate the understanding of the domain expert. Beside proposing a meta model and a graphical language, we developed a REA-DSL tool as a proof of concept.

In this work, we concentrated on the basic REA principles [4] and the value chain perspective [13]. In future work, we will gradually extend the REA-DSL. Currently, we are working on an integration of the REA policy infrastructure [18] covering commitments, agreements and, furthermore, the typification of the operational concepts [5]. Next, we plan to extend the REA-DSL by concepts to derive a database design for enterprise information systems, which has been one of the original goals of REA. For this purpose, we have already introduced the concepts of economic event series and economic resource series in the current DSL, because they will affect the multiplicities in the database design. By including the policy infrastructure and the REA-driven database design, it will become more obvious that REA models are of structural nature rather and do not concentrate on the behavioral aspects of process models. Another intended REA-DSL extension addresses the perspective from which the REA models are described. Currently, we focus on the perspective of a single enterprise which ex-

changes value with other enterprises. In the Open-edi reference model (ISO/IEC 15944-4) REA concepts are used to describe the exchanges of value among enterprises from a neutral observer's point of view. We plan to integrate the observer's perspective into our REA-DSL and to support semi-automatic mappings between the perspectives.

References

1. Andersson, B., Bergholtz, M., Edirisuriya, A., Ilayperuma, T., Johannesson, P., Gordijn, J., Grégoire, B., Schmitt, M., Dubois, E., Abels, S., Hahn, A., Wangler, B., Weigand, H.: Towards a Reference Ontology for Business Models. In: Proc. of the 25th Int. Conf. on Conceptual Modeling, Springer LNCS (2006) 482–496
2. Osterwalder, A., Pigneur, Y., Tucci, C.L.: Clarifying Business Models: Origins, Present and Future of the Concept. Communications of the Association for Information Science (CAIS) **15** (2005) 751–775
3. Gordijn, J., Akkermans, H.: E3-value: Designing and evaluating e-business models. IEEE Intelligent Systems **16**(4) (Jul–Aug 2001) 11–17
4. McCarthy, W.E.: The REA Accounting Model: A Generalized Framework for Accounting Systems in a Shared Data Environment. The Accounting Review **57**(3) (1982)
5. Geerts, G.L., McCarthy, W.E.: An ontological analysis of the economic primitives of the extended-rea enterprise information architecture. International Journal of Accounting Information Systems **3**(1) (2002) 1 – 16
6. van Deursen, A., Klint, P., Visser, J.: Domain-specific languages: an annotated bibliography. SIGPLAN Not. **35**(6) (2000) 26–36
7. Kelly, S., Tolvanen, J.P.: Domain-Specific Modeling. Wiley-IEEE Computer Society Press (2008)
8. Greenfield, J., Short, K., Cook, S., Kent, S.: Software Factories. John Wiley (2008)
9. Strembeck, M., Zdun, U.: An approach for the systematic development of domain-specific languages. Softw., Pract. Exper. **39**(15) (2009) 1253–1292
10. Geerts, G.L., McCarthy, W.E.: An Accounting Object Infrastructure for Knowledge-Based Enterprise Models. IEEE Intelligent Systems **14**(4) (1999) 89–94
11. Geerts, G.L., McCarthy, W.E.: The Ontological Foundations of REA Enterprise Systems (August 2000)
12. Motal, T., Schuster, R.: From e3-value to REA: Modeling multi-party eBusiness Collaborations. In: Proc. of the 11th IEEE Conference on Commerce and Enterprise Computing, IEEE CS (2009) 202–208
13. Geerts, G.L., , McCarthy, W.E.: Modeling business enterprises as value-added process hierarchies with resource-event-agent object templates. In: In Business Object Design and Implementation, Springer-Verlag (1997) 94–113
14. OMG: Meta Object Facility (MOF) Core Specification, Version 2.0 (January 2006)
15. Ijiri, Y.: Theory of accounting measurement. American Accounting Association, Sarasota, Fla (1975)
16. Yu, S.C.: The Structure of Accounting Theory. The University Presses of Florida (1976)
17. Gailly, F., Poels, G.: Ontology-driven business modelling: Improving the conceptual representation of the rea ontology. In: Conceptual Modeling - ER 2007, Springer LNCS (2007) 407–422
18. Geerts, G.L., , McCarthy, W.E.: Policy-level specification in rea enterprise information systems. Journal of Information Systems **20**(2) (2006) 37–63