Paired Transactions and Their Models

Frantisek Hunka(✉) and Jiri Matula

University of Ostrava, Dvorakova 7, 701 03 Ostrava, Czech Republic
{frantisek.hunka, jiri.matula}@osu.cz

Abstract. Paired transactions or paired transfers have their origin in accountancy systems. Resource-Event-Agent (REA) ontology uses paired transactions as a basic building block for business process modelling. A business process (REA model) is composed of two kinds of paired transactions which stand for provide (give) and receive (take) transactions (transfers). The REA core pattern which comes from double-booking entry, constitutes the basic structure from which the REA model is further extended. The REA model was previously depicted by ER diagrams, and later by UML class diagrams. However, neither of these diagrams were designed to capture conceptual models which are more precise and comprehensible for domain experts, and easy to modify. ORM (Object Role Modeling) is a Fact-Based Modeling methodology that represents an approach to conceptual modelling that fulfils the requirements. The article presents fact-based models of paired transactions corresponding to the REA core pattern and REA exchange model. These models were created and verified by NORMA (Natural ORM Architect) and provide both semantic stability and semantic relevance, which is necessary for further incorporating the REA modeling approach into fact-based modeling methodologies of business processes.

Keywords: REA ontology · Paired transactions · Fact-based modelling · ORM method

1 Introduction

REA (Resource-Event-Agent) ontology can be classified as a domain-specific ontology that focuses on value-modelling of business processes. The term value-modelling, in this context, means that the REA modelling approach keeps track of primary and raw data about economic resource values. Economic resources can be exchanged for other economic resources within the scope of REA exchange processes or they can be consumed, used or produced within the scope of REA conversion processes. The three core REA concepts are *resource*, *event* and *agent*, from which the name of the modelling approach was derived. The aim of the REA modelling approach is to record any changes in property rights to resources, record resource usage, record resource consumption, or resource production, see [1, 6, 9].

The REA model records information based on the coherence between data of one or more economic events. The REA process is defined by related economic events and has at least two composite economic events: a *decrement event* that outflows, consumes or uses the outgoing resource(s), and an *increment event* that inflows or produces the

E. Ziemba (Ed.): AITM 2016/ISM 2016, LNBIP 277, pp. 20–34, 2017.
DOI: 10.1007/978-3-319-53076-5_2

incoming resource(s). The REA process is called the *REA model* and represents the notion of a business process. The main benefit of the REA modelling approach is the possibility of keeping track of primary and raw data about economic resources, by [7]. All accounting artefacts such as debit, credit, journals, ledgers, receivables and account balances are derived from data describing exchange and conversion REA processes [6, 10]. For example, data describing a sale event is used in warehouse management, payroll, distribution, finance, and other application areas, without transformations or adjustments. This explains why the REA approach can offer a wider, more precise and more-up-to-date range of reports. The REA ontology also benefits from the presence of a semantic and application independent data model, an object-oriented perspective and abstraction from technical and implementation details [7, 9]. These features enable the possibility of calculating the value of the enterprise's resources on demand, as opposed to calculation at pre-determined intervals.

The quality of a database application depends essentially on its design, see [15]. To ensure correctness, clarity, adaptability and productivity, information systems should be specified at the conceptual level first, using concepts and language that both designers and customers can easily understand [8]. Object-Role-Modeling (ORM) is a fact-oriented approach for modelling information at a conceptual level. A fact is a particular arrangement of one or more objects. Depending on the number of objects that are involved in a fact, we speak of unary, binary, ternary, etc., facts. An example of unary fact is a *Vendor is a Person*. An example of a binary fact is that a *Customer receives a Pizza*. Unlike traditional approaches, ORM make no use of attributes such as base constructs, instead expressing all fact types as relationships [8]. This attributes-free-approach leads to greater semantic stability in conceptual models, and enables ORM fact structure to be directly verbalized and populated using natural language sentences.

ORM methodology provides a more precise way to capture and validate data concepts and business rules with domain experts. This methodology provides higher semantic stability and semantic relevance than ER and UML diagrams. Semantic stability is a measure of how well models or queries expressed in the language retain their original intent in the face of changes in the business domain, according to [8]. Semantic relevance requires that only conceptually relevant details need to be modelled, by [8]. ORM also focuses on structural changes in the application. By omitting the attribute concept, ORM allows communication in simple sentences. ORM diagrams simply capture the world in terms of objects (entities or values) that play roles (parts in relationships) which forms fact. ER notation, as well as UML notation, allow relationships to be modelled as attributes. ORM models that capture the world in terms of objects and roles have only one data structure – the relationship type. As a consequence, ORM diagrams take up more space than corresponding UML or ER diagrams. The aim of the paper is to apply the ORM modelling approach to REA core patterns and REA models to obtain easily comprehensible, verifiable, and easily implementable solutions.

This paper is an extended version of the shorter and less comprehensible contribution to the Federated Conference on Computer Science and Information Systems (FedCSIS), which was held in Gdansk in 2016. The extension covers the Fact-Based Model of the REA core pattern, a more precise description and explanation of the

Fact-Based Model of the REA model, and a verification of both models. Apart from these main features the whole text has been revised, corrected, and modified in order to increase comprehensibility and overall clarity.

The structure of the paper is as follows: An overview of business process modelling is provided in Sect. 2. Section 3 describes the REA modelling approach. A concise description of the possibilities of the ORM modelling method are mentioned in Sect. 4. The ORM model of the REA core pattern is described and illustrated in Sect. 5. Section 5 illustrates and depicts the ORM model of the REA model. Discussion of the results is featured in Sect. 6, and the conclusion is contained in Sect. 7.

2 Business Process Modeling

There are a number of methods that deal with business process modelling. Currently the most important methods are IDEF0; see [16], Business Process Modeling Notation, flowcharts, use cases, and data models. The approaches mentioned above model business processes using general-purpose concepts such as activities, data entities, etc., with or without poorly defined rules for formulating well-formed models of enterprise processes. These methods usually cannot offer an answer to questions such as how activities serve to increase the value of an enterprise's resources, and, consequently, these models cannot answer the question of why the enterprise performs its activities.

Research initiatives in the domain of accounting information systems have resulted in proposals of new data models in accounting, especially in models focusing on the modelling of the resource value. New modelling and system design techniques are required for information technologies that can support the enterprise in achieving and sustaining the necessary flexibility. However, traditional process models do not display resource control and value flows. This is the domain particular to value modelling ontologies. Currently, the most popular approaches within the area of enterprise ontologies are e3-value; see [17], and REA ontology for enterprise processes; see [1, 2].

The e3-value ontology stipulates that the actors exchange value objects by means of value activities. The value activity should yield a profit for the actor. Deeper insight into e3-value modelling (such as [17], shows that this method only covers the exchange and trade processes and omits production and conversion processes. The state-of-the-art e3-value model only focuses on the operational level (what has happened), but not on management policies (what could or should happen).

REA ontology links together business process modelling with underlying economic phenomena. It benefits from the presence of a semantic- and application-independent data model, an object-oriented perspective, and abstraction from technical and implementation details.

The REA model is the first semantic application-independent data model which utilizes an object-oriented perspective. The REA model originates from the domain of accounting; see [4] and matured to a conceptual framework and ontology for Enterprise Information Architectures; see [1, 3]. The REA model focuses on core economic phenomena and abstracts from technical and implementation details. The REA model provides concepts to store past and future data consistently.

On the other hand, REA ontology anomalies have their origin in the absence of a rigorous theoretical foundation. The REA model itself does not have given states from which a state machine can be derived. Instead, only the resource states are identified and frequently used as the states of the state machine. As a result, the REA model does not provide revoking operations such as cancellation and roll-back mechanisms. In addition, the REA model is predominantly designed to capture events that are concerned with resource values or resource features. Other events such as business events or information events are difficult to capture and process further. Consequently, REA has difficulty with so-called information or knowledge entities such as contract or schedule. It is possible to create them because creation indicates a value. But the real problem lies in determining the state for which the information entities are valid, see [10].

3 The REA Modeling Approach

The REA core pattern, which comes from double booking entry, underpins the fundamental part of REA ontology [4, 6]. It was McCarthy [4] who was first to recognize that an enterprise's economic activities follows the REA core pattern in which causally related *provide* (give) and *receive* (take) economic events are associated with resources and agents. The fundamental entities of the pattern are different events that are involved in various transactions which have something in common; there has always been a *decrement economic event* (one in which something is provided) and an *increment economic event* (one in which something is received). In addition to economic events, the economic agents that represent human beings partake in the exchange process. Resources are entities which are tracked because their property rights can be exchanged or they can be converted to create new resources. As mutually reciprocal events cannot happen at the same time, the claim entity is utilized for deferred revenue, prepaid expenses, accounts payable, and so on. The REA core pattern is illustrated as a Pizzeria shop in Fig. 1. Economic events in Exchange processes represent the permanent or temporary transfer of property rights to economic resources from one economic agent to another. The transfer of property rights represents the increment or decrement of the value of resources. That is why the term 'value' is utilized in the name of the *value-modelling* approach. The economic events in REA models usually encapsulate properties for *date*, *time* and *location* in space.

Economic events, and the duality relationship by which events are related, play a crucial role in this pattern. All other entities that participate in this pattern are related 'through' economic events. Both agents are related by increment and decrement events because they lose rights to given resources and gain rights to other resources. On the other hand, resources are related to corresponding economic events. The relationship between increment and decrement events is exchange duality. The purpose of exchange duality is to keep track of which resources were exchanged for which. In the REA value model of an exchange process, every increment economic event must be related by an exchange duality to a decrement economic event, and vice versa.

The REA model is an extension of the REA core pattern. The principal feature of the REA modelling approach is that it explicitly distinguishes between past and current events and events performed in the future for which it introduces the commitment

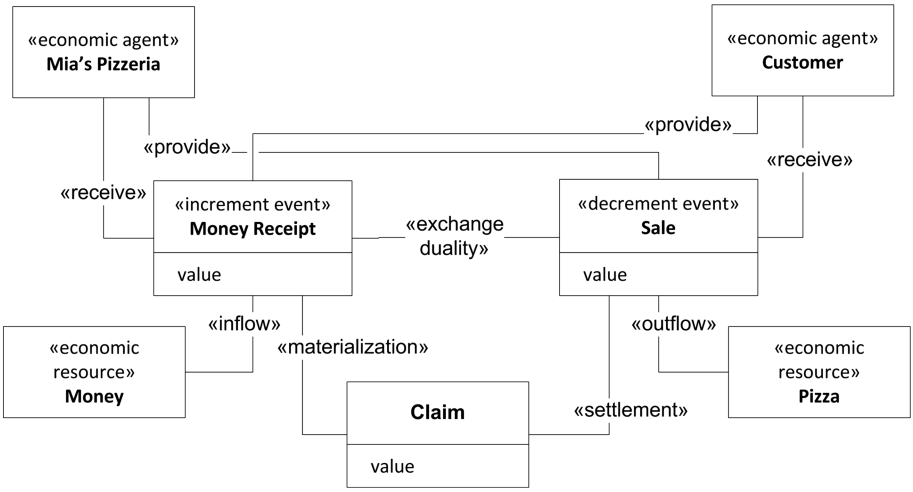

Fig. 1. The REA core pattern example

entity. This explicit distinction between these types of events has its origin in the progressive development of the REA Framework. Curiously, the utilization of a commitment entity is not obligatory but depends only on specific modelling circumstances. The REA core pattern is much easier and more straightforward to use for modelling than the REA model. When utilizing the REA core pattern, a database solution contains only real values of resources involved in transactions without planning events, which makes this approach incomplete.

The relationships of *committed provide* and *committed receive* mean that some agreement about the future exchange must be achieved between economic agents. The commitment entity addresses the issue of modelling promises of future economic events and the issue of reservation of resources. Commitment entities and their relationships with other entities are shown in Fig. 2. In this case, Fig. 2 is an extension of Fig. 1. To a considerable extent, the commitment entity copies the structure of the event entity, by which we mean the existence of an increment and decrement commitment and the exchange reciprocity relationship. The exchange reciprocity relationship between the increment and decrement commitments identifies which resources are promised to be exchanged for which other resources. The reciprocity relationship is one of many-to-many.

Each commitment is related to an economic resource by a reservation relationship which specifies which resources will be needed or expected by future economic events. The reservation relationship between the resource and commitment represents the obligation of economic agents to provide or receive rights to economic resources in exchange processes, and represents scheduled usage, consumption or production of economic resources in conversion processes.

The most important relationships of the REA model are the *exchange reciprocity* and *exchange duality* relationships, by [7, 11–13]. The exchange reciprocity relates a pair of increment and decrement commitment entities. The exchange reciprocity

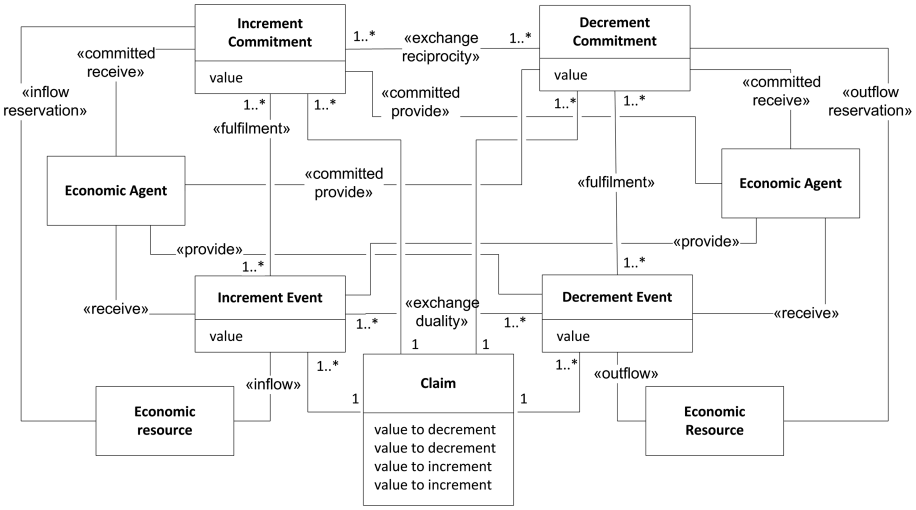

Fig. 2. REA model. Adapted from [7]

relationship identifies which resources are promised to be exchanged for which others. The exchange duality relationship, which relates corresponding increment and decrement economic events, keeps track of which resources were exchanged for which others.

4 ORM Conceptual Modeling Method

Object-Role Modeling (ORM) is a conceptual modelling method that ranks among fact-based modelling methodologies. ORM views the world as a set of objects that play roles (parts in relationships) according to [8]. For example, an individual may play the role of walking in the country (a unary relationship involving just the individual) or an individual may play a role reading this paper (a binary relationship between the individual and the paper). Thus a role in ORM corresponds to an association-end in UML, except that ORM also allows unary relationships. Object-Role Modeling is a conceptual modelling method that views the world as a set of objects that play roles (parts in relationships) according to [8].

The main structural difference between ORM and UML is that ORM excludes attributes as a base construct and treats them instead as a derived concept. The conceptual schema using ORM specifies the information structure of the application in the forms of: *fact types* that are of interest; *constraints* on these fact types; and *derivation rules* for deriving some other facts.

A fact is a proposition that is taken to be true by the relevant business community. A fact type is a kind of fact that may be represented in the database [5]. The constraints represent constraints or restrictions on populations of fact types. The derivation rules include rules that may be used to derive new facts from existing facts, see [8, 14].

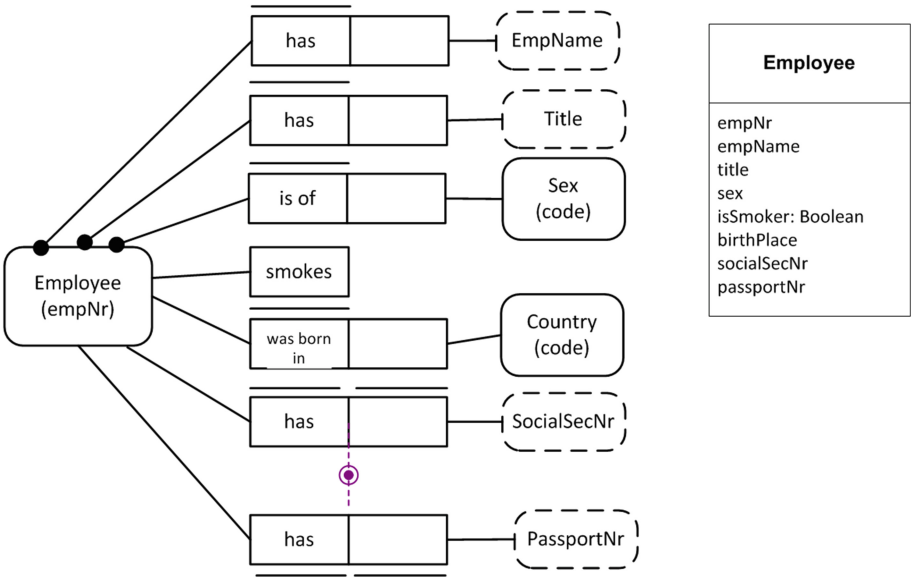

Fig. 3. ORM and UML model of Employee

The ORM model (left part of Fig. 3) indicates that employees are identified by their employee numbers. The top three roles (*EmpName, Title* and *Sex*) are mandatory roles. This is indicated by the black dots at the *Employee* box. The circled mandatory role dot depicts an *inclusive-or* constraint over the roles hosted by the Employee in the fact type. This indicates that an employee must have a social security number or a passport number or both. The uniqueness of constraints (cardinalities in UML) is indicated by vertical lines over roles. In Fig. 3 it means that *empNr, EmpName, Sex* and *Country* is unique for each employee. Two vertical lines over each roles (*SocialSecNr, PassportNr*) indicate that each employee number, social security number and passport number refers to a maximum of one employee. The dashed line over e.g. *PassportNr* indicates that this is a value not an object.

Graphically, object types are depicted as named boxes (solid for entity types, and dotted for value types). As in logic, a predicate is a proposition with object-holes in it. In ORM, a predicate is treated as an ordered set of one or more roles, each of which is depicted as a box, which may optionally be named. A fact type is formed by applying a predicate to the object types that play its roles. Fact type *roles* are depicted as *role boxes*, connected by a line segment to the object type that hosts the role.

In a similar way to UML class diagrams, ORM methodology also enables the use of sub-typing mechanism to classify object types. It enables specialization of some instances of an object type into more specific types. For example, people (persons) at a university may be roughly classified as staff and students.

5 ORM Model of the REA Core Pattern

The REA core pattern which was depicted in Sect. 2 comes from double-booking entry and deals with only current and past economic events. It is fully in compliance with the functional needs of double-booking entry. Figure 4 displays the ORM diagram of the REA core pattern. The left-hand side transactions represent *transfers of goods*, and the right-hand transactions stand for *transfers of money*. We consider the common example of a transfer, in which resources or services are exchanged for money.

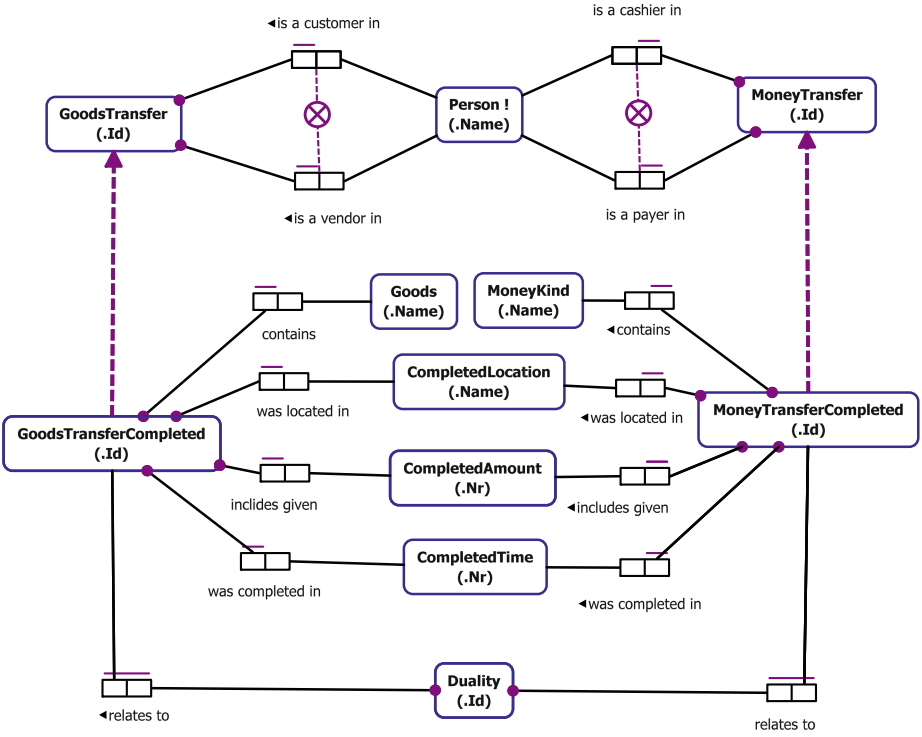

Fig. 4. ORM Model of the REA core pattern

Named boxes in the figure represent object types - more specifically entity types or value types. The object types *Duality*, *Goods Transfer*, *Money Transfer* and *Person* are primal types, and are not defined on the basis of other object types. Figure 4 indicates that the object type *Duality* has a relationship to the object types *Goods Transfers* and to the object type *Money Transfers*. The position of uniqueness constraint (vertical line over the roles) indicates that a *Duality* can be determined by one or more *Goods Transfer types*, as well as by one or more *Money Transfer types*. Goods may be delivered by several shipments and payments may be performed by several instalments, as in reality. The black dots at the *Duality* box indicate that the roles of *Duality* are mandatory roles. It means that the type *Goods Transfer* or *Money Transfer* may exist

independently before the existence of the type *Duality*, and that the object type *Duality* must have s relationship to *Goods Transfer* and to *Money Transfer*.

The object type *Goods Transfer* has a relationship to the role of *vendor* and *customer*. Both roles belong to the object type *Person*. *Exclusive-or* relationship between two actor's roles means that these actor roles must be filled by different persons. The customer is an actor role that receives the goods in *Goods Transfer*. The *vendor* is an actor role that provides the goods for the transfer. Similarly, the corresponding object type *Money Transfer* is related to the payer and *cashier* actor's roles. Between both actor roles there is *exclusive-or* relationship with the same meaning as in the case of a *customer* and a *vendor*. Both the *payer* and *cashier* belong to the object type *Person*. It is important to use a proper actor role. The *customer* can be, for example, a wife and the payer can be her husband.

The object type *Goods Transfer Completed* is a sub-type of *Goods Transfer*. Sub-typing helps to organize the way we think about objects in the world. The sub-type relationship means that any goods transfer can become a goods transfer completed. So, goods transfer can enter the phase of being completed. The expression of this kind of specialization perfectly suits modelling requirements. Each object type *Goods Transfer Completed* also hosts the roles that the object type *Goods Transfer* hosted. These are, namely, the roles *Vendor* and *Customer*.

The object type *Goods Transfer Completed* has a relationship to the entity type *Goods* and is identified by the value types *Amount*, *Time* and *Location*. These entity and value types completely fulfil practical needs for *Goods Transfer Completed*. *Goods* in this case represents concrete instance of the entity type, the value type amount expresses measurable (quantifiable) units of goods. The value type *Time* indicates time specification of delivery, and, finally, the value type *location* is a concrete place specification. The black dots at the *Goods Transfer Completed* box indicate that the roles of *Goods Transfer Completed* are mandatory roles. It also indicates that the entity type *Goods* and the value types *Amount*, *Time*, and *Location* may have existed before *Goods Transfer Completed* became existent. Mandatory roles hosted by all object types in Fig. 4 indicate that attributes may exist before the existence of the object types. But if *Goods Transfer Completed* becomes existent, its entity types and value types must already exist.

The object type *Money Transfer* forms reciprocal transaction to the transaction *Goods Transfer*. The object type *Money Transfer Completed* is a sub-type of *Money Transfer*. Rationale for the object types and the roles hosted by the object types is very much the same as in the case of the object type *Goods Transfer Completed.*

REA model has also been modelled and verified by using NORMA (Natural ORM Architect). A human-readable definition of the model is automatically generated via verbalization browser integrated in NORMA. Such a method helps to discover inaccuracies in the model and allows to populate model with test data even before the actual implementation of the database application. This emphasizes the considerable expressivity of the ORM approach compared to ER diagram models. Here follows the example (Fig. 5).

The REA model being an extension of the REA core pattern, the ORM model of the REA model is also an extension of the ORM model from the previous section. The ORM model of the REA model is also composed of two kinds of transactions (left

Duality **is an entity type**.
Reference Scheme: Duality has Duality_Id.
Reference Mode: .Id.
Data Type: Numeric: Auto Counter.

Fact Types:
Duality has Duality_Id.
GoodsTransferCompleted relates to Duality.
Duality relates to CompletedMoneyTransfer.

Examples:

MoneyTransfer is paid by Person.
Each MoneyTransfer is paid by **exactly one** Person.
It is possible that some Person paid **more than one** MoneyTransfer.

Examples:
MoneyTransfer MTRN001 is paid by Person Joseph.

MoneyTransfer is accepted by Person.
Each MoneyTransfer is accepted by **exactly one** Person.
It is possible that some Person accepted **more than one** MoneyTransfer.

Examples:
MoneyTransfer MTRN001 is accepted by Person Phoenix.

CompletedMoneyTransfer includes MoneyKind.
Each CompletedMoneyTransfer includes **exactly one** MoneyKind.
It is possible that some MoneyKind is a part of **more than one** CompletedMoneyTransfer.

Examples:
CompletedMoneyTransfer CMTRN001 includes MoneyKind 'EUR'.

Person is a customer in GoodsTransfer.
For each GoodsTransfer **exactly one** Person is a customer in **that** GoodsTransfer.
It is possible that some Person is a customer in **more than one** GoodsTransfer.

Examples:
Person John is a customer in GoodsTransfer TRN001
GoodsTransferCompleted relates to Duality.
Each Duality is related to **exactly one** GoodsTransferCompleted.
It is possible that some GoodsTransferCompleted relates to **more than one** Duality.

Fig. 5. The example part of verbalized REA core pattern model in ORM

hand side, right hand side). Each kind of transaction is composed of three phases. The first initial phase and the final completed phase come from the ORM model of the REA core pattern.

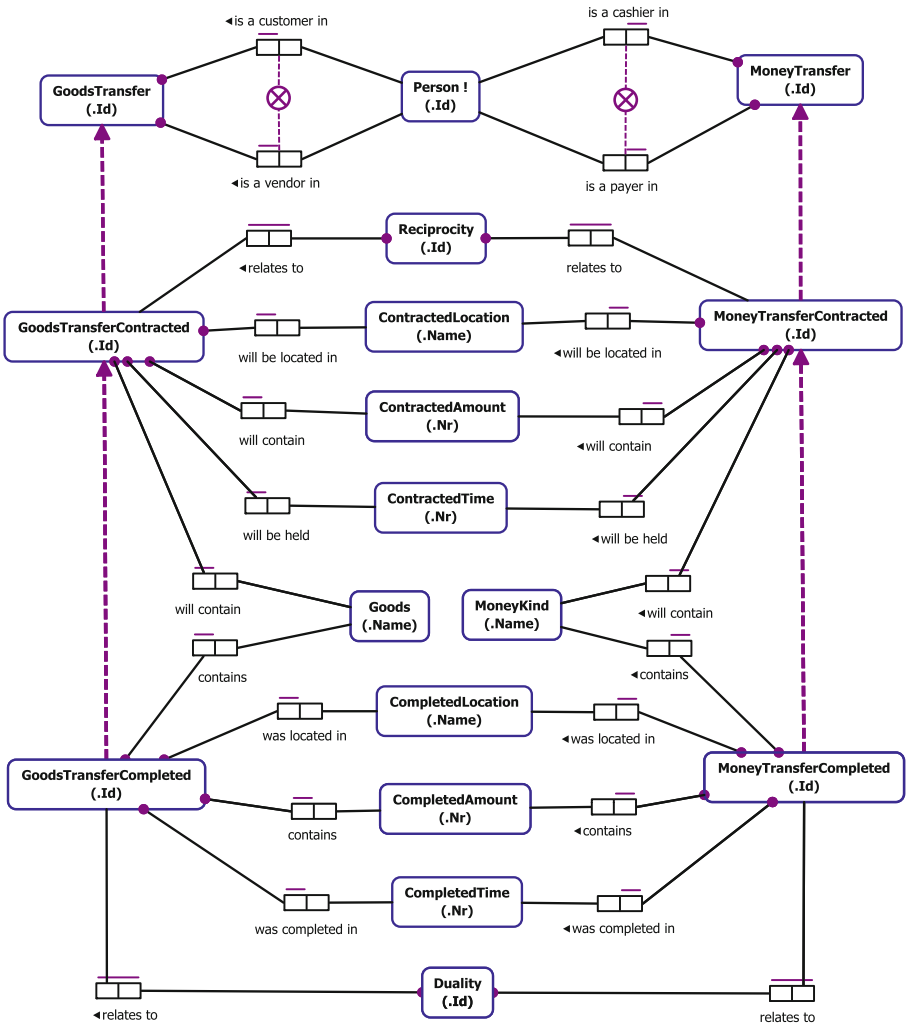

Fig. 6. ORM model of paired transactions (REA model)

The contracted phase is a new phase and is represented by the object types *Goods Transfer Contracted* and *Money Transfer Contracted*. This phase stands for future (planned) events (in REA notation – commitment). The ORM model of the REA model is shown in Fig. 6.

The ORM model of the REA model is extended by the object types *Goods Transfer Contracted* and *Money Transfer Contracted*. At first, we focus on a description of how these new object types change the overall structure of the model. The object type *Goods Transfer Contracted* is a direct sub-type of the type *Goods Transfer*. Next, the object type *Goods Transfer Completed* is a direct sub-type of the object type *Goods*

Transfer Contracted. It indicates that *Goods Transfer* first enter the phase of being contracted, and then enter the phase of being completed. In short, any *Goods Transfer* can become *Goods Transfer Contracted*, but only a contracted goods transfer can become a completed goods transfer.

The structure of the object type *Money Transfer* is similar to the *Goods Transfer* structure. The object type *Money Transfer* plays a fundamental role in this kind of transaction. The object type *Money Transfer Contracted* is a direct sub-type of *Money Transfer*, and the object type *Money Transfer Completed* is a direct sub-type of *Money Transfer Contracted.* By analogy with the previous paragraph, any *Money Transfer* can become *Money Transfer Contracted*, but only a contracted money transfer can become a completed money transfer.

The object types *Goods Transfer Contracted* and *Money Transfer Contracted* mutually express that the agreement specifying goods and their delivery, as well as the amount of money and conditions of payment, as reached. More specifically, the *customer* enters into an agreement about goods and their specification and the *vendor* promises to deliver the stated goods by certain time, in a certain location. The *payer* promises to pay according to the agreement, and the *cashier* promises to accept the money. Unless an agreement is reached, the exchange process will not continue. In order to express this condition explicitly, the object type *Reciprocity* that has a relationship to the object types *Goods Transfer Contracted* and to *Money Transfer Contracted* must be utilized. Its mandatory roles indicate that both *Goods Transfer Contracted* and *Money Transfer Contracted* have become existent.

Each object type *Goods Transfer Contracted* and *Money Transfer Contracted* has a relationship to contracted resource kinds (kind of goods and kind of money) and is identified by value types further specifying the entity type resource. The value types are *amount*, *time* and *location*. The structure of these entity and value types is the same as in the case of *Goods Transfer Completed* (*Money Transfer Completed*) but the meaning is different. These entities and value types represent planned values or kinds of resources. In reality, they may differ from the entity and value types stated at *Goods Transfer Completed* (*Money Transfer Completed*). On the other hand it is desirable to record both data concerning the contracted phase of the exchange process, as well as data concerning the completed phase of the exchange process.

The other transaction(s) of the paired transactions is represented by the object type *Money Transfer Contracted*. In this case, the payer has promised that he will pay for the goods and the cashier has promised that he will accept the amount of money (resource). At this point it is essential that the state of promise is achieved on both object types *Goods Transfer Contracted* and *Money Transfer Contracted* which is checked and validated by the object type *Reciprocity*.

6 Discussion of Findings

Designing an information system involves building a formal model of the application domain which requires a good understanding of the application domain and utilization of the proper methodology for modelling specifications in a clear and unambiguous way. ORM simplifies the design process by using natural language, and by examining

the information in terms of simple and elementary facts. By expressing the model in terms of natural concepts, such as objects and roles, it provides a conceptual approach to modelling comprehensible also to domain experts.

The REA core pattern is directly based on double booking entry and captures modelling reality accurately. It is fairly comprehensible, but its drawback is its inability to model future events (commitments). However, this is brought about by the modelling paradigm (double-booking entry), which addresses only past and current economic activities. In addition, a temporally Claim entity is utilized to balance time differences between fulfilling transaction kinds.

The underlying object types of the ORM model of the REA core pattern are *Goods Transfer* and *Money Transfer.* These object types form the initial phase of the exchange process. *Goods Transfer Completed* is a derived sub-type from *Goods Transfer,* and *Money Transfer Completed* is a derived sub-type form *Money Transfer*. Derived sub-types create a completed phase of the exchange process. The sub-typing mechanism allows the introduction of state and state transitions. In reality, it means that first *Goods Transfer* becomes existent and then *Goods Transfer Completed* can occur. The same is also valid for *Money Transfer* and *Money Transfer Completed*. The crucial entity in the REA core pattern is the *Duality* entity that keeps different kinds of transactions together. In the ORM model of the REA core pattern, the object type *Duality* has a relationship to *Goods Transfer Completed* and to *Money Transfer Completed*. This is because this object type relates transactions in the completed phase. In other words, *Duality* indicates that both kinds of transactions have been successfully terminated. This approach enables the elimination of the entity type *Claim* from the ORM model.

Contrary to the REA core pattern, the REA model represents a solution which captures future events as an inseparable part. This is in compliance with reality, and not only with the requirements of double-booking entry. However, REA relationships between the commitment entity and event entity suffer from a number of deficiencies. Firstly, both entities are too separated from each other. They should create an integral unit, yet they do not. Secondly, they suffer from the absence of an REA transaction oriented state machine. The current REA state machine addresses the states of resource or resource type in question. Finally, these entities formally belong to different modelling levels; economic commitments belong in the policy level and economic events have their place at the operational level. A solution is rather difficult to implement.

The ORM model of the REA model represents the ultimate solution. The sub-typing mechanism is utilized to separate different phases of the exchange process. More specifically, it is used to discern the initial phase from the contracted phase and from the completed phase. Phase's design is intuitive and comprehensible. The contracted phase is a new phase which is added into the phases of the ORM model of the REA core pattern. It is important phase because this phase signifies the obligation to fulfil future economic events. The value types and entity types which identify both contracted transfers represent planned (promised) value types and entity types. Entity types represent a resource specification and value types represent an amount, time and location specification. Both in reality and in modelling, it is important to precisely determine whether the different kinds of transactions were contracted. For this reason, the object type *Reciprocity,* which has a relationship to both kinds of transactions, was introduced.

The object types *Goods Transfer* and *Money Transfer* form the beginning of the paired transactions process, which means that they identify partaking actor roles which will be involved in the process. They must be created first. After this, the object types *Good Transfer Contracted* and *Money Transfer Contracted* can be created. Mutual creation of these object types is checked and verified by the object type Reciprocity. Similarly, existence of *Goods Transfer Completed* and *Money Transfer Completed* is dependent on the existence of contracted transfers. Mutual existence of these object types is checked and verified by the object type *Duality*.

The ORM model which is exhibited in Fig. 6 is composed of 24 fact types. Mandatory roles, uniqueness constraints, and the sub-typing mechanism make the model comprehensible and complete.

7 Conclusion

The article presents fact-based models of paired transactions corresponding to the REA core pattern and REA exchange model utilizing the ORM conceptual modeling method. Presented models were created and verified by NORMA (Natural ORM Architect). They provide both semantic stability and semantic relevance, which is necessary for further incorporating the REA modeling approach into fact-based modeling methodologies of business processes.

The benefits of applying this method on paired transactions is mainly in distinction of the initial phase, the contracted phase, and the completed phase of the paired transactions ORM model and in support of unified transaction processing. The individual phases can be seen as the states of the REA exchange process.

The REA modeling approach is to some extent inherently confined to the changes in the value of economic resources. This aspect negatively affects creation of the complex REA concepts such as contract or schedule. On the other hand, mapping elementary facts delivered by a generic modeling methodology to the complex REA concepts could be an effective solution to the problem. This is the aim of further research.

Acknowledgements. The paper was supported by the grant provided by the Ministry of Education, Youth and Sport Czech Republic, reference no. SGS15/PRF2016.

References

1. Geerts, G.L., McCarthy, W.E.: The ontological foundation of REA enterprise information systems. In: Paper Presented at the Annual Meeting of the American Accounting Association, Philadelphia (2000)
2. Geerts, G.L., McCarthy, W.E.: Policy-level specifications in REA enterprise information systems. J. Inf. Syst. **20**(2), 37–63 (2006)
3. Geerts, G.L., McCarthy, W.E.: Using object-oriented templates from the REA accounting model to engineer business process and tasks. In: Paper Presented at European Accounting Congress, Gratz, Austria (1997)

4. McCarthy, W.E.: The REA accounting model: a generalized framework for accounting systems in a shared data environment. Acc. Rev. **57**, 554–578 (1982)
5. Dietz, J.L.G.: The Essence of Organization, 2nd edn. Sapio Enterprise Engineering (2015). http://www.sapio.nl
6. Dunn, C.L., Cherrington, O.J., Hollander, A.S.: Enterprise Information Systems: A Pattern Based Approach. McGraw-Hill/Irwin, New York (2004)
7. Hruby, P.: Model-Driven Design Using Business Patterns. Springer, Heidelberg (2006)
8. Halpin, T.: Object-role modeling fundamentals. A practical guide to data modeling with ORM. Technics Publications, New Jersey (2015)
9. Dudycz, H., Korczak, J.: Conceptual design of financial ontology. In: Ganzha, M., Maciaszek, L., Paprzycki, M. (eds.) Proceedings of the Federated Conference on Computer Science and Information Systems, FedCSIS 2015, pp. 1505–1511. Polskie Towarzystwo Informatyczne, IEEE Computer Society Press, Warsaw, Los Alamitos (2015)
10. Hunka, F., Zacek, J.: Detailed analysis of REA ontology. In: Aveiro, D., Tribolet, J., Gouveia, D. (eds.) EEWC 2014. LNBIP, vol. 174, pp. 61–75. Springer, Heidelberg (2014). doi:10.1007/978-3-319-06505-2_5
11. Hunka, F., Zacek, J.: A new view of REA state machine. Appl. Ontology **10**(1), 25–39 (2015)
12. Klimek, R., Szwed, P.: Verification of archiMate process specification based on deductive temporal reasoning. In: Ganzha, M., Maciaszek, L., Paprzycki, M. (eds.) Proceedings of the 2013 Federated Conference on Computer Science and Information Systems, FedCSIS 2013, pp. 1103–1110. Polskie Towarzystwo Informatyczne, IEEE Computer Society Press, Warsaw (2013)
13. Korczak, J., Dudycz, H., Dyczkowski, M.: Design of financial knowledge in dashboard for SME managers. In: Ganzha, M., Maciaszek, L., Paprzycki, M. (eds.) Proceedings of the 2013 Federated Conference on Computer Science and Information Systems, FedCSIS 2013, pp. 1111–1118. Polskie Towarzystwo Informatyczne, IEEE Computer Society Press, Warsaw (2013)
14. Kersten, G., Wachowicz, T.: On winners and losers in procurement auctions. In: Ganzha, M., Maciaszek, L., Paprzycki, M. (eds.) Proceedings of the 2014 Federated Conference on Computer Science and Information Systems, FedCSIS 2014, pp. 1163–1170. Polskie Towarzystwo Informatyczne, IEEE Computer Society Press, Warsaw (2014)
15. Paweloszek, I.: Approach to analysis and assessment of ERP system. A software vendor's perspective. In: Ganzha, M., Maciaszek, L., Paprzycki, M. (eds.) Proceedings of the 2015 Federated Conference on Computer Science and Information Systems, FedCSIS 2015, pp. 1415–1426. Polskie Towarzystwo Informatyczne, IEEE Computer Society Press, Warsaw, Los Alamitos (2015)
16. IDEF0, Integration definition for function modeling. National Institute of Standards and Technology, FIPS Publication (1993). http://www.idef.com/pdf/idef0.pdf
17. Gordijn, J., Akkermans, H.: Value based requirements engineering: exploring innovative e-commerce idea. Requirements Eng. J. **8**(2), 114–134 (2003)