REA-ERP: Challenges of using REA in an ERP System

Dieter Mayrhofer[1], Christian Huemer[1], Peter Regatschnig[2]

[1]TU Vienna, [2]eventus Marketingservice GmbH
[1]{mayrhofer, huemer}@big.tuwien.ac.at, [2]peter.regatschnig@eventus.at

Abstract. The Resource-Event-Agent (REA) ontology is a powerful and well accepted approach towards the design of enterprise information systems in the academic world. However, it still lacks application in industry products. Software like ERP applications might benefit from REA data structure by building their product's core on a robust, economic theory based, business ontology. Consequently, we have lately started a feasibility study with an Austrian ERP vendor, in which we investigate whether or not the ERP vendor can use the REA ontology as basis for their new ERP system. This paper is aimed at raising questions about uncertainties in REA we came across in our study so far.

1 Introduction

The Resource-Event-Agent (REA) ontology developed by McCarthy, Geerts and others [1] is the most prominent business ontology for accounting information systems. REA is a widely accepted framework for the design of conceptual models of the accountability infrastructure of enterprise information systems. In its beginnings, REA described the resource flows within and between companies capturing what is currently occurring and what has occurred in the past. This is known as the operational layer. Later REA was extended by a planning layer and a policy layer capturing what should, could, or must be occurring sometime in the future [2, 3]. Today, REA may be considered as a powerful business ontology capturing all relevant data to generate the conceptual design of an Accounting Information System (AIS) as well as of an Enterprise Resource Planning (ERP) system.

During the last 30 years, REA has achieved wide acceptance in the academic world but still lacks implementation in industrial products. To the best of our knowledge, there are no maintained ERP systems based on REA available, which are used by companies. Thus, it is still unknown if REA can live up to the requirements of an ERP system. Furthermore, REA instance examples are rare, which may help database designers in a great extent to understand the concepts behind REA.

In order to assess these uncertainties we lately started conducting a feasibility study with an Austria based ERP vendor, who considers to use REA for the data structure of a revamped generic ERP system. We are right now in an early state of sketching the ERP data structure according to REA. However, there are already questions arising from this task concerning the proper use of REA. Furthermore, performance concerns may arise when using REA in its intended way. We would like to use these questions and concerns for an open discussion in this paper. We believe, that we can point out some areas of REA which need more research and are crucial for industrial acceptance.

In the following, we talk about the major uncertainties we stumbled upon so far: (i) positions in commitments, (ii) performance with identifiable resources in commitments/events, (iii) negotiation, and (iv) unhappy paths.

2 Uncertainties

Uncertainty 1: Is each position in one separate commitment or are they one combined commitment? In REA, a binding contract is established if two parties commit to exchange resources in the future. This contract consists of at least two commitments, which are in reciprocity to each other (cf. Figure 1). For example, in an order one party `Agent A` commits to provide two specific resources `Prod 1` and `Prod 2` to party `Agent B`. In return (reciprocity), party `Agent B` commits to pay a certain amount of money to party `Agent A`.

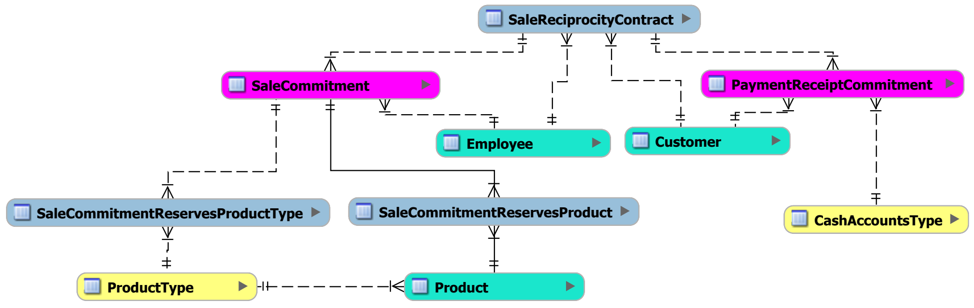

Fig. 1. Commitment relational model

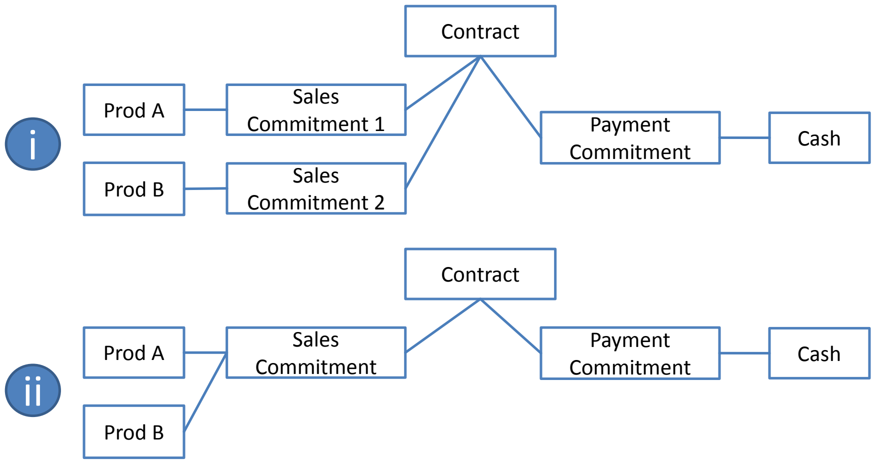

Fig. 2. Commitment instance

When looking at REA instance examples (cf. Figure 2), we see two possibilities: (i) one commitment for each resource `Prod 1` and `Prod2` or (ii) one commitment combining both resources. Is REA meant to capture each position in a separate commitment as shown in (i)? This would mean, that the customer legally commits to each single position instead of committing to the whole order at once as shown in (ii). This also leads to many more rows in the database which may cause worse database performance.

Uncertainty 2: If multiple identifiable products of the same type are reserved through stockflows in one commitment/event, is it allowed to additionally save the cumulated quantity as a product type stockflow? When looking at Figure 3, the usual way to reserve identifiable resources is (i) by reserving each of them (Prod A1 to A500). Imagine someone ordering 500 products of the same type, this would take some considerable time to query the complete order, where I am not interested in each single instance of the 500 products (i.e., each single serial number). Thus, we would propose additionally to the stockflows of the identifiable products (ii) one cumulated stockflow with a quantity of 500 to the product type. Consequently, when displaying an order, we do not have to query over all 500 products. Instead, it is sufficient enough to query over all product type stockflows, in this case just one. In other cases, I can however still query the serial numbers of the products in the order.

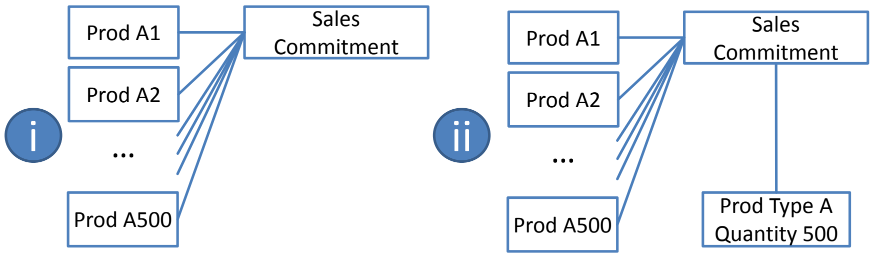

Fig. 3. Commitment reserving products and product type

Uncertainty 3: Negotiation - how to handle negotiation in REA? Let's assume following example: `My company` sends out a RequestForQuote for 10 specific candy bars (there is no legal commitment yet). `Three companies` reply to this RequestForQuote with a quote (`company A` 10 Euro, `company B` 15 Euro, and `company C` 20 Euro). Thus, they commit themselves to deliver the candy bars for the stated money. We now pick `company A` and send them a counter offer which says, that we would order the candy bars for 8 Euro (consequently, we commit to this price). `Company A` accepts this counter offer. Now, `my company` and `company A` both committed to transfer 8 Euro for 10 candy bars.

In REA, we have a reciprocity between two commitments, namely OrderCommitment and PaymentCommitment. Does it make sense to reflect the negotiation process by updating the commitment instance in each negotiation step (which will not store the history of the negotiation process in that case)? Or should we introduce an additional entity in the REA model which handles the negotiation process and also allows to reveal the history of the negotiation process? Satoshi and McCarthy [4] have presented some work on introducing state machines for the negotiation process of commitments. However, there is still no common agreement how to handle negotiation in REA.

Uncertainty 4: How to deal with unhappy paths and complaints (e.g., product return)? Many papers and textbooks on REA consider the "happy path". "Unhappy paths" are mentioned rarely and often in simple ways. Lets assume we have sent two cameras to a customer. With one of the two cameras the customer is not satisfied because he or she believes it is not working properly. In REA literature, this case is often solved by a product return: the customer returns the product and we return the cash. Figure 4 shows one possible simple data structure which allows to support this case. However, in real world this case can be more complicated. Here are some scenarios: (i) the product is returned and the money is returned, (ii) the product is returned and we ship a new one, (iii) the product is returned but we figure out that it is still working and resend the same unit to the customer, (iv) the product is returned and we put credit on the customers account, (v) the product is not returned and we give the customer a discount, (vi) the product is not returned and the customer just pays one product and we do not collect the payment for the second product, (vii) the second product was a wrong delivery caused by us, (viii) ....

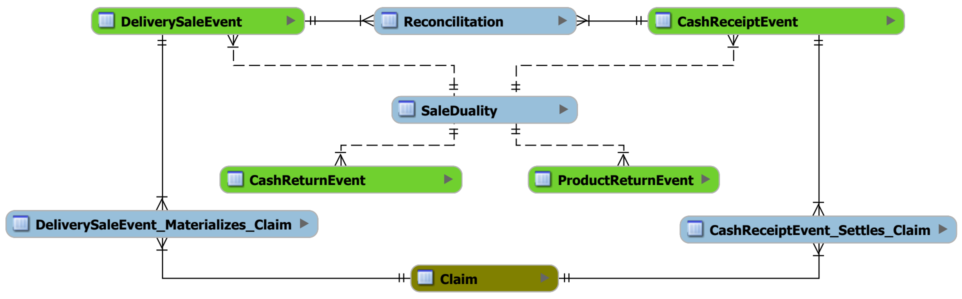

Fig. 4. Complaints

You see, that there are many ways how a complaint can be resolved. In a generic ERP system, we have to consider all of these cases. A case may comprise of various events which affect the claims and the duality. Furthermore, how can we model the intention of a customer to return a product? Is REA mighty enough to capture the history of these complaints? Sometimes it is a subjective matter if a duality is reconciled or not, how should we handle these cases?

3 Conclusion

We are in the first month of our feasibility study and came across those four uncertainties mentioned above among others. We hope, that this paper is a start of a broader discussion on how to make REA more suitable for industry and real world scenarios. We believe that a reference model for REA purchase and sale business activities can be the basis for such discussions and might help many ERP/IS database designers to get started with REA.

References

1. McCarthy, W.E.: The REA Accounting Model: A Generalized Framework for Accounting Systems in a Shared Data Environment. The Accounting Review **57**(3) (1982)
2. 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
3. Geerts, G.L., , McCarthy, W.E.: Policy-Level Specification in REA Enterprise Information Systems. Journal of Information Systems **20**(2) (2006) 37–63
4. Horiuchi, S., McCarthy, W.E.: An Ontology-Based State Machine for Catalog Orders. In: Proceedings of the 5th International Workshop on Value Modeling and Business Ontologies (VMBO 2011), Ghent, Belgium. (2011)