A Process Interpretation of REA models

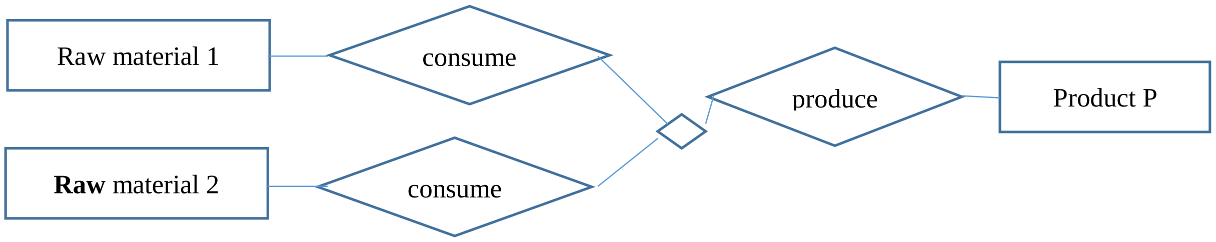
Hans Weigand, Birger Andersson, Maria Bergholtz, Paul Johannesson
This is a work-in-progress, we are looking forward to feedback from the VMBO community

The Resource-Event-Agent (REA) ontology was first formulated in (McCarthy, 1982) and has been developed further, e.g. in (Geerts, McCarthy, 2006) and (Hruby, 2006). The following is a short overview of the core concepts of the REA ontology.

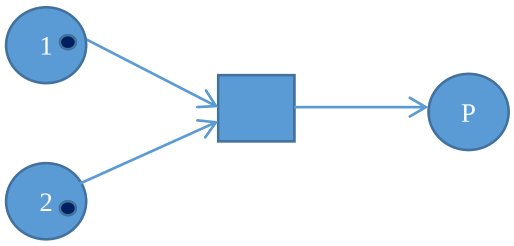
In REA, a *resource* is any object that is under the control of an agent and regarded as valuable by some agent. Resources are modified or exchanged in processes. The constituents of processes are called *economic events*. REA recognizes two kinds of duality between events: conversion duality and exchange duality. A *conversion process* consumes some input resources to produce new or modify existing resources, like in manufacturing. An *exchange process* occurs as two agents provide/receive resources that flow out or into the company. A fundamental principle is that to acquire a resource (increment event) an agent has to give up some other resource (decrement event). REA is a suitable value model as it abstracts from process details and implementation systems, and focuses on economic *value*. It has been shown that REA structures also provide a basis for auditing (Weigand, Elsas, 2012). The dualities express fundamental integrity constraints that can be used for both the design of control mechanisms (preventive) and for the detection of deviating behavior (detective) that may indicate fraud.

In the original REA paper, REA was introduced as an accounting framework for use in a shared data environment and developed on top of ER modeling (Chen, 1978), so with a clear focus on database design. However, events play a key role as atomic data units. Given the presence of events in the REA models, it is also possible to give a dynamic interpretation. In this paper, we want to show how REA models can be interpreted as Petri Nets. This is not a completely original idea. For instance, Laurier and Poels (2013) show a mapping from REA to Petri Nets (Expect) and how these Petri Nets can support useful business simulations. Others have combined REA with dynamic models (state machines, DEMO transaction models, ..) in order to model the business processes or transactions in which REA events are coordinated - processes that the REA ontology as such is abstracting from. The goal of this paper is to give a dynamic interpretation to REA *itself*, and investigate some possible applications.

Let us start with a REA conversion event. We depict decrement events (out stockflow) as diamonds with <consume>, and increment events (in stockflow) as diamonds with <produce>. Resources are depicted as rectangles. The duality (white diamond) links N outflow to M inflow events. Agents could be added easily but are not considered in this paper.

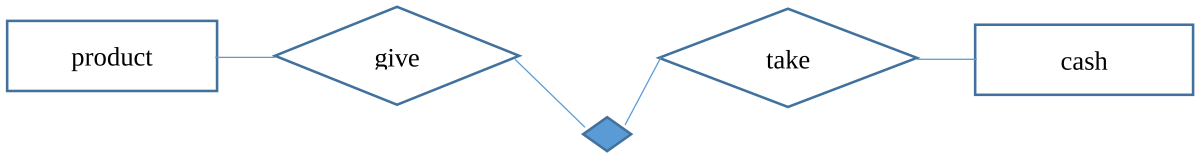


This model corresponds directly to the following Petri Net:

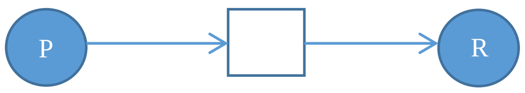


The meaning of this Petri Net is that once there is a token in place1 and place2 (the black dots), the transition can fire and an event is executed. This means that the tokens in place1 and place2 are *consumed* and a token in place P is *produced*. The two consumptions correspond 1-1 with the outflow events in the REA model, and the token production corresponds 1-1 with the inflow event. The places in the Petri Net correspond to buffers. Note that the interpretation of the REA resource in the dynamic model is that it represents a *buffer* of resources of some type, e.g. an inventory. Consumption stockflow events decrease the inventory level and production stockflow events increase the level. The REA conversion duality states that production and consumption are in balance – you cannot have one without the other. On a conceptual level this means in Petri Net terms that each transition must have input and output places (at least one of each). More precisely, the duality takes the form of some conservation law. For instance, mass in (kg) = mass out (kg). In practice, the conservation law is an empirically grounded equation relating input and output, e.g. $1.P \approx A.RM1 + B.RM2$, that says, that for the production of one P element you need *A* RM1 and *B* RM2 elements, where A and B are coefficients. Typically, there is some variance in the conversion process, so the equation is an idealization. It can be used to check the plausibility of actual event data (e.g. in auditing), in simulations or cost accounting. The coefficients could be added to the REA model similar to the way cardinality and modality of relationship is specified in ER models, but with the difference that they have an stochastic value.

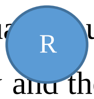
Now we consider a REA exchange duality. Here there seem to be more possible interpretations.

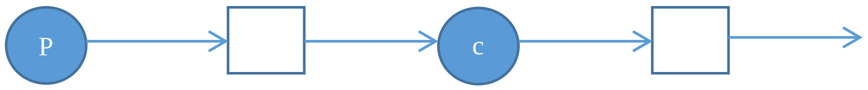


A direct Petri Net interpretation looks as follows:

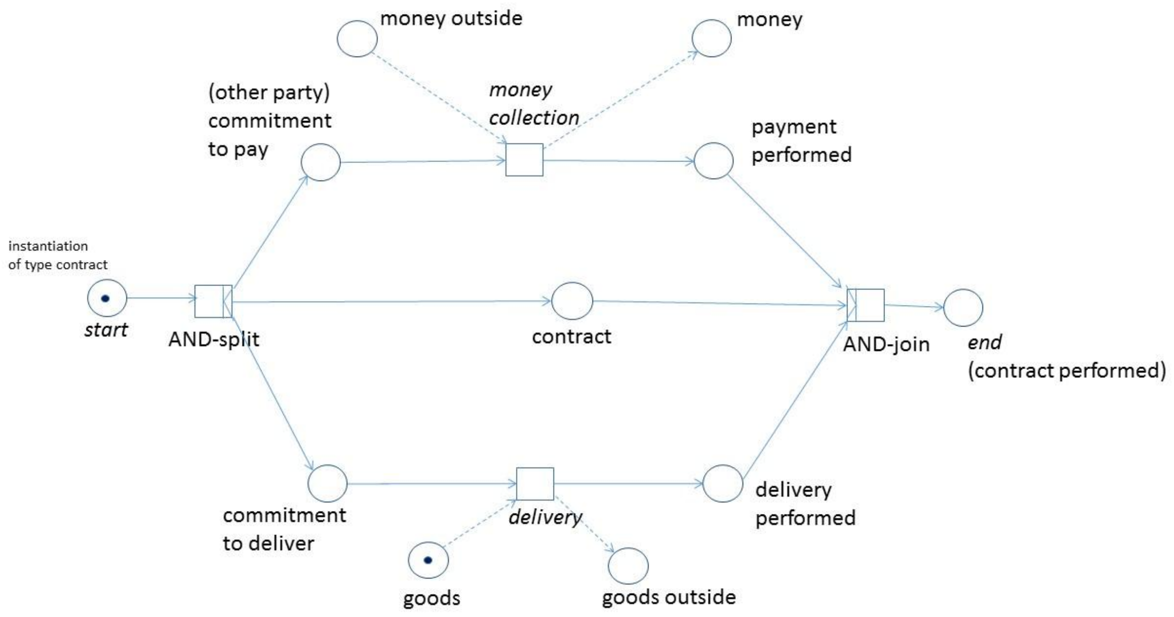


Revenue (money) is produced in a transition in which product is consumed. One cannot be done without the other. This is a correct implementation of the exchange dua[illegible]ut does not take into account the time delay that may exist between the inflow and outflow and the uncertainty that is introduced by the involvement of an external agent (the customer). Therefore REA sometimes also includes claims. A delivery produces a claim c that is resolved once the payment is received (revenue produced), or when it is written off as uncollectable.



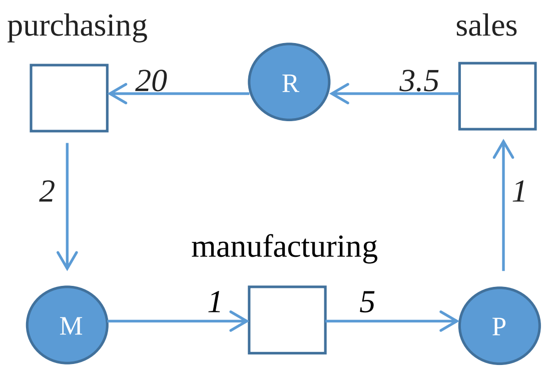


A more general treatment of the exchange process represents it as a workflow that starts with the instantiation of a contract. This is followed by three parallel lines, one for payment, one for delivery and one for the contract itself:



Now payments can be received any time after the contract has been instantiated, before or after the product has been delivered. At the end, the AND join closes the case when there is agreement between delivery, contract and payment. At this point, the exchange duality gets realized. The basic pattern could extended with extra constraints, for instance, to enforce that delivery is only enabled after payment. The extended model can be seen as a subprocess that introduces some extra places – these are part of the state of the Petri Net and as such are relevant for the annual account. The balance sheet typically gives the value of all places at the time, like stock, claims, and not only revenue.

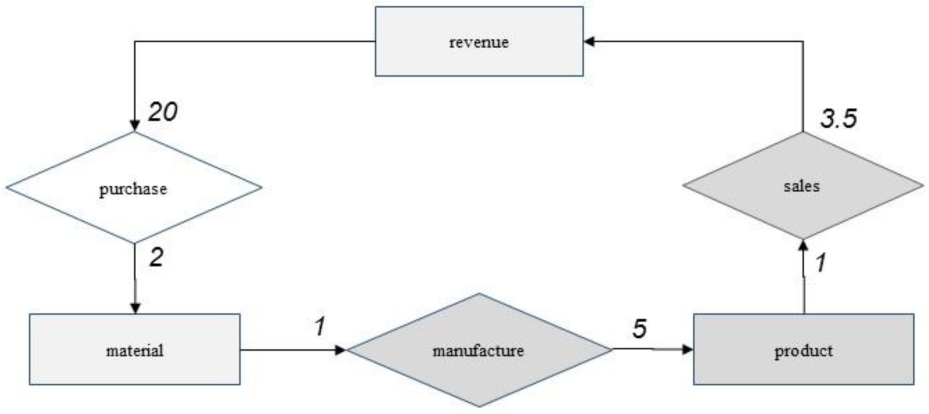
Taking the simplest representation for the exchange duality, the basic structure of the complete value cycle is as follows. We have added coefficients.



Typically value is created in this cycle (value jump) as sales price is higher than cost price – this is the *profitability* constraint. The numbers in the schema indicate the coefficients. So for 20 money units 2 material units are purchased. From each material unit, 5 products can be produced, and each product unit is sold for 3.5 money units. So 20 money units in the start state lead to 10 products and 35 money units produced. No resources are left in the other places and the value jump is 15 (money units). In this simple example, the only costs are the purchasing costs of the material, but other costs like human resource hours can be added.

Graphically, the same cycle can also be represented in ER style. Now the diamonds represent processes (REA dualities – instances thereof), that get a unique id in the database, and each line can be said to represent a REA event, where the arrow indicates the stockflow type. When mapping this model to a data base schema, the events must be mapped to tables. The rectangles represent resources/entities; they are mapped to object identifiers (individual or generic). The event table typically contains an event id, an object id (the resource affected), the process id (so that the event can be related to dual events), an agent id and the stockflow quantity and time stamp. The object id of the resource can be seen as master data (it does not change during transactions): additional information about objects includes a full name, a maximum capacity, a location etc.

For an actual company, the REA model is much more complicated of course, but it can be decomposed into a set of value cycles. Our claim is that this representation is easy to understand, comprehensive and more informative (because of the coefficients) than other current representations, and as complete (although the REA events are depicted implicitly).



Model quality

Once the REA model is interpreted as a Petri Net, standard quality features such as Liveness, Connectedness and Boundedness can be defined. Evidently, a value cycle is not bounded. Liveness is a desirable property, but the question is how critical it is for REA models. In contrast to business process models, coordination is not what REA modeling addresses. Still, the completeness of the model can be important from a Resource Planning perspective and in Cost Accounting.

Audit applications

Elsas (1996) was one of the first to apply Petri Net models for audit analysis. Using formal analysis fraud scenarios can be discovered that are not detected (with the given internal controls). Once REA models are given a Petri Net interpretation, these audit analysis techniques can be applied in REA as well.

References

Elsas, P.I. (1996) Computational Auditing, Ph.D. thesis, Free University, Amsterdam; with a contemporary introduction http://www.slideshare.net/PhilipElsas/siks-elsas-final2, presented at the 2010 SIKS Smart Auditing PhD course http://www.siks.nl/SA-2010.php

Geerts, G., McCarthy, W. (2006) Policy-Level Specifications in REA Enterprise Information Systems. Journal of Information Systems, Vol. 20 Issue 2, pp. 37-63

Hruby, P. (2006) Model-Driven Design of Software Applications with Business Patterns. Springer

Laurier, W., & Poels, G. (2013). Invariant conditions in value system simulation models. *Decision Support Systems*, *56*, 275-287

McCarthy W.E. (1982) The REA Accounting Model: A Generalized Framework for Accounting Systems in a Shared Data Environment. The Accounting Review, pp.544-577

Weigand, H., Elsas, Ph. (2012) Model-based Auditing using REA. Int. Journal on Accounting Information Systems, Vol. 13, No. 3, pp. 287-310