Transforming the Resources/Events/Agents Model into a Formal Process-Oriented Enterprise Framework

Christian Stefansen

Department of Computer Science, University of Copenhagen (DIKU)
Universitetsparken 1, DK-2100 Copenhagen Ø
Denmark

Abstract. This position paper presents the author's personal and unratified views on the Resource/Events/Agents model (REA) along with suggested directions of further research with an emphasis on formal methods. The three main suggestions for REA are: (1) to establish a formal foundation, (2) to design a contract/process language, and (3) to design a report language.

1 Introduction

The Resources/Events/Agents model (REA) [McC82,GM00,GM02] provides an enterprise system ontology rooted in the microeconomic theory of the firm and in particular the notion of *exchanges*.

Thus, its original purpose was to capture economic phenomena, but the model has the potential of covering a much wider area including business process modeling and contract modeling. Modeling and executing business processes in one integrated model that traverses functional silos as well as organizational borders is a long-held dream among AIS researchers. In my view this requires an augmented REA model that has

1. a sound formal foundation,
2. a strong contract/process language, and
3. an efficient report language.

Each point is elaborated in turn in the sections below. Possible future uses of an augmented REA model could be:

1. automated reasoning about processes (i.e. checking for non-termination, deadlocks, livelocks, boundedness, etc.)
2. valuation and optimization of processes
3. automated contract execution/monitoring
4. automating processes across supply-chains
5. (semi-)automatically derived workflow GUIs (cf. [DKKS04])

2 A Formal Foundation

The most important prerequisite of a formal foundation is a well-specified model. The issues discussed in this section should be resolved before the model can be properly formalized.

The ontology is a trifle too abstract. More specifically, its distinction between types and instances it slightly unclear, cardinalities between base objects are not properly defined, and a thorough analysis of the trade-offs between size and fit of the ontology is missing (cf. [BS04]). REA ontology work has primarily been theory-driven as opposed to other models that are the product of alternating between theoretical work, implementation, and case studies [KZR04]. This perhaps is the reason that it is hard to find a unique operational interpretation of the REA model. The model is more flexible than it might otherwise have been, but allows too much room for personal interpretation. In essence, I feel there is a need for an operational REA model with a broad consensus among REA researchers.

Another object of discussion is the exchange pattern. It is very strong in forcing designers to apply an Activity-Based Costing ([ABKY01]) mindset, but if in the future we want to model business processes using information events, it is not obvious whether the constraint is desirable or not. Moreover, it is somewhat unsettling that the axioms (defined as fundamental truths to which there are no counter-exampels) of the REA ontology can be violated in "implementation compromises"—such ad hoc designs potentially impede interoperability. In my view it is worth considering if the duality axiom should be demoted to a "best practice" or something similar.

A formal model should preferably be as simple as possible. Therefore, a discussion of the three-level architecture might prove fruitful:

1. Is it needed? Can one general level/sublevel pattern model the same in a simpler way?
2. Has it been used in practical settings? What are the results?

3 A Contract/Process Language

The REA model's notion of *agreement* does not include conditionals, predicates, or temporal aspects. An obvious path of research is to embed Peyton Jones' idea of combinator contract languages [JES00] into the REA model.

Another approach is to fit Petri nets or an appropriate process calculus to the REA model. These formalisms are well-known and bear with them a substantial body of research. A number of attempts already exist in the area of modeling workflows using these techniques [vdAvH02,JMM$^+$99]. Both the combinator approach and the process calculus approach deserve some research attention and ideally a comparison.

Peyton Jones' language does not model agents (parties) explicitly. Based on the nature of REA, it seems reasonable to make agents explicit. This immediately gives rise to a need for parameterized contracts. For some contracts, agents may be variable, while for other contracts agents may be fixed while the resources involved vary. A natural solution would be lambda-abstractions. This allows stepwise specialization of a contract, which would essentially behave like a curried function.

Another generalization is predicates. Rather than having combinators that limit the time horizon Peyton Jones-style, a general predicate language could be plugged in. In this way any contract could be constrained to a more limited contract by putting it inside a predicate combinator.

Also, there is a general need for use cases in this area. What contracts are commonplace? What supply-chain scenarios should be supported? What production schedules should be supported? Should information back-propagation be encoded using information events? What common workflows should be supported? How do policies tie in with a more complex contract language?

4 A Report Language

The REA model only specifies first-order entities. Procedural aspects and aggregation for reporting is left as design decisions for implementors. A language or method for aggregating such data should be specified more precisely. This will pave the way for algorithmic research on efficiency (using e.g. finite differencing [PK82]).

As with contracts, use cases are needed. What financial reports are needed? What reports are needed to support a diverse range of costing systems?

5 A side-note: Mapping the Double-Entry Ledger to REA

Mapping the double-entry ledger to REA is reasonably straightforward, bar one thing: owner's equity. Modeling owner's equity explicitly is in a sense the epiphany leading to double-entry bookkeeping. Owner's equity is an accounting artifact and thus not an explicit part of the REA model. It is unclear how to express accruals, dividends, capital adjustments, and more arcane equity destributions in REA. It would be very helpful to have this sorted out, e.g. by a handbook in REA bookkeeeping. Also, such a handbook would provide very useful when selling the REA idea.

References

[ABKY01] Anthony A. Atkinson, Rajiv D. Banker, Robert S. Kaplan, and S. Mark Young. *Management Accounting*. Prentice Hall, third internation edition edition, 2001.

[BS04] Signe Ellegård Borch and Christian Stefansen. Evaluating the REA enterprise ontology from an operational perspective. Submitted to INTEROP '04., March 2004.

[DKKS04] Nikunj P. Dalal, Manjunath Kamath, William J. Kolarik, and Eswar Sivaraman. Toward an integrated framework for modeling enterprise processes. *Communications of the ACM*, 47(3):83–87, 2004.

[GM00] Guido Geerts and William E. McCarthy. The ontological foundation of REA enterprise information systems, August 2000.

[GM02] Guido Geerts and William E. McCarthy. An ontological analysis of th economic primitives of the extended-REA enterprise information architecture. *International Journal of Accounting Information Systems*, (3):1–16, 2002.

[JES00] Simon Peyton Jones, Jean-Marc Eber, and Julian Seward. Composing contracts: an adventure in financial engineering (functional pearl). In *Proceedings of the fifth ACM SIGPLAN international conference on Functional programming*, pages 280–292. ACM Press, 2000.

[JMM$^+$99] Wil Janssen, Radu Mateescu, Sjouke Mauw, Peter Fennema, and Petra van der Stappen. Model checking for managers. In *SPIN*, pages 92–107, 1999.

[KZR04] Rajiv Kishore, Hong Zhang, and R. Ramesh. A Helix-Spindle model for ontological engineering. *Communications of the ACM*, 47(2):69–75, 2004.

[McC82] William E. McCarthy. The REA accounting model: A generalized framework for accounting systems in a shared data environment. *The Accounting Review*, LVII(3):554–578, July 1982.

[PK82] Robert Paige and Shaye Koenig. Finite differencing of computable expressions. *ACM Transactions on Programming Languages and Systems*, 4(3):402–454, July 1982.

[vdAvH02] Wil van der Aalst and Kees van Hee. *Workflow Management: Models, Methods, and Systems*. MIT Press, 2002.