An accounting object infrastructure for knowledge-based enterprise models

Guido L. Geerts and William E. McCarthy
Michigan State University

The infrastructure of a corporate enterprise information system is concerned with acquiring, transferring, converting, and selling economic resources such as cash, inventory, and supplies. Accounting systems with individual computerized modules—such as payroll, accounts payable and receivable, job costing, order entry, and general ledgers—have traditionally tracked such data, and accounting system designers usually have had a preemptive call on the basic schemes used to type economic data. This is because of a need to meet either statutory reporting requirements for governmental agencies, or private reporting requirements for creditors and shareholders.

If accountants use these preemptive privileges to force traditional account coding onto the organization database as its fundamental classification architecture (that is, to require early bookkeeper filtering of transaction data), then many dysfunctional effects arise.[1,2] Prominent among these effects are an inability to

- accommodate the process-oriented models of the enterprise,
- integrate well with knowledge-based decision models of other enterprise domains such as supply-chain management and strategic decision making, and
- support interorganizational use.

Our remedy for these deficiencies is to apply the REA (economic *r*esources, *e*vents, and *a*gents) conceptual model when analyzing, designing, implementing, and operating an enterprise information system. (For more information, visit http://www.reavillage.org.) REA specifically incorporates the semantics of economic objects into a firm's information architecture. Such embedding facilitates an information system's initial design as well as couples its production use with knowledge-based decision-support systems.

Recent advances in REA theory[3,4] have also incorporated its explicit top-down use as a Michael Porter-type[5] process model of enterprise economic activities along a value-added chain. These extensions let the firm view the process semantics of its constellation of economic objects at multiple levels of abstraction, and the firm can use these semantics as an explicit domain conceptualization, both within the firm and between the firm and its trading partners. The extended REA structures also let a firm build a semantic enterprise model that links to other initiatives, such as the introduction of business-process reengineering and workflow management or the movement toward activity-based management.

REA accounting as a script

In starkly simple terms, all business enterprises operate in the same manner. Somebody has an idea about how to provide a new or improved service or product. This entrepreneur acquires some initial financing (debt or equity for the enterprise), then engages in a chain of economic exchanges with other parties (such as vendors and employees)—each time giving up an economic resource (perhaps money) in return for another resource of greater value. Value is defined as a deliverable portfolio of product or service attributes attractive to the firm's ultimate customers.

Hopefully, most entrepreneurs find that when they have consummated their final exchanges with customers and paid their creditors, they enjoy a justifiable profit. A successful entrepreneur continually cycles through such a chain of value-added activities. A corporation does the same, except on a larger scale and in a more bureaucratic fashion.

Figure 1a illustrates this entrepreneur script at different levels of abstraction. The top-level process, "Engage in value-added exchanges," is exploded to the three second-level processes, each identifying economic resources as both input and output. Accountants refer to these three second-level processes as the

- *acquisition* cycle—cash is exchanged for labor and raw materials,
- *conversion* cycle—labor and raw materials are converted into finished goods, and
- *revenue* cycle—finished goods are exchanged for cash.

A firm's REA process model can consider such a process hierarchy at great depths, but here we show just two levels. A typical enterprise model would have a much deeper and wider process hierarchy that would approximate the full Porter value chain. (In an earlier work, we explained the microeconomic rationale for these enterprise models.[4])

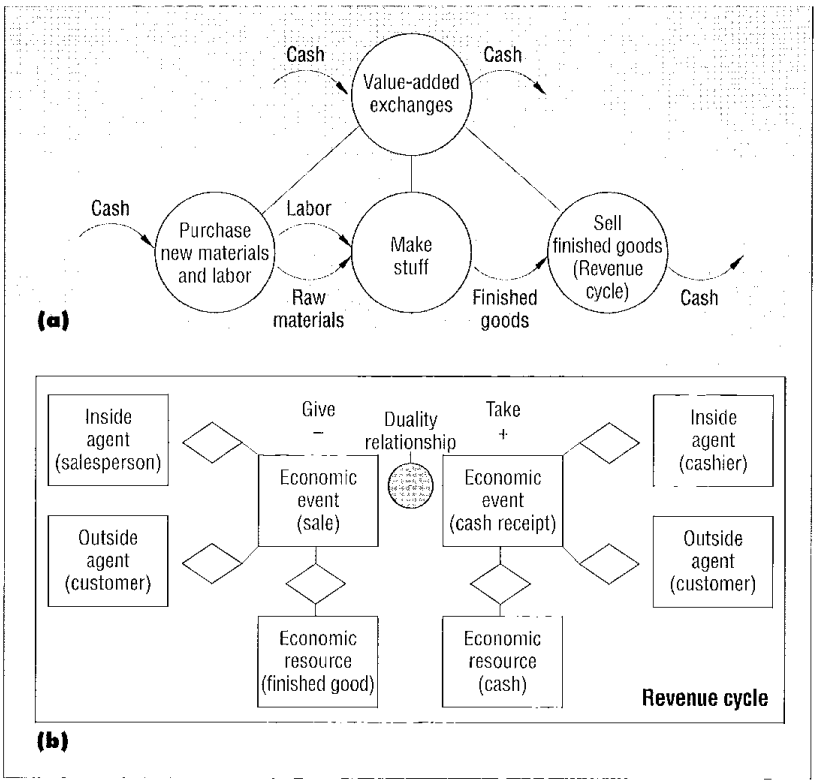

Figure 1. REA entrepreneur script: (a) the business entrepreneur script and (b) the give-and-take constellation of entities that each value-added exchange (revenue cycle) experiences.

Figure 1b shows the REA object constellation (in entity–relationship form) of every business process. In general, each process includes eight entities, although there is often overlap or aggregation. Each exchange has an increment event (or possibly a set of events) linked with a decrement event (or set of events). The increment and decrement events have entity constellations or patterns that mirror each other.[1]

In very general terms, an REA process has a *give* (a resource or a set of resources consumed) paired with a *take* (a resource or set of resources acquired). These gives and takes are connected by duality relationships. To make matters more concrete, Figure 1b notes that the revenue cycle (selling the finished goods) might have the following set of REA entities:

- decrement: a sale (event) occurs, which involves a salesperson (inside agent) giving the finished goods (resource) to a customer (outside agent); and
- increment: a cash receipt (event) occurs, which involves a cashier (inside agent) taking the cash (resource) from a customer (outside agent).

The activity decomposition of an enterprise typically produces a process hierarchy much deeper and wider than the two levels Figure 1a illustrates. More realistically, these processes would be decomposed to the representation level at which management must plan, control, and evaluate[6] actual economic events before they are detailed in the object structures of Figure 1b. Additionally, each cycle typically would also include other less prominent resource decrements (such as a salesperson's labor) called *transaction costs* in the process representation. However, we stress simplicity in our explanations here, so we omit these decompositions and additional decrements from the figure.

REA-based architectures

When the entrepreneur script is fully specified top-down and when each leaf node in the process hierarchy is examined to give its full complement of REA entities and relationships, the result is a company's candidate enterprise schema. Taking some of the REA objects' *type images*[4] can expand this candidate schema. For example, in the revenue cycle we discussed, these types might include different employee segments for skill deployment, different customer segments for marketing purposes, and different inventory categories for profitability analysis. However, in all corporate cases, this enterprise schema must be augmented with many objects not directly related to acquiring, converting, and selling economic resources.[7]

Nonetheless, the REA components will undoubtedly form an accountability infrastructure for the corporate information architecture, which includes most of the objects needed to manage the firm (see Figure 2). The actual technology platform for implementing this architecture could include semantically designed databases, object-oriented systems, or even traditional legacy systems that have been wrapped with object representations. The only requirement for their knowledge-intensive use is that the object semantics of the REA value chain be made explicit.

Figure 2 shows both the process and economic object flavor of an REA infrastructure. The five processes each have their appropriate economic events portrayed, although space constraints preclude delineation of the economic resources and agents. Note that this model shows value-added processing occurring from left to right. Additionally, we can further divide each of the illustrated process events into a set of the tasks needed to accomplish them. At this task level, an REA model shows the workflow elements such as the data communication between departments and the ordering of manual and computer process steps.[8] The task level is where most reengineering efforts take place in companies.

Figure 2 also illustrates the transaction input and output that would be associated with day-to-day operation and use of a knowledge-intensive enterprise information system based on our explanations thus far. This input and output can be seamlessly integrated with that of other upstream and downstream partners in the value chain who have made the same ontological commitment to REA-based data definitions. Most of the concept population in the firm's object structure would come from this transaction-level input. However, other nontransaction sources could be used systematically in an integrated fashion. These might include both managerial information and estimates from inside the firm (such as budget information or engineering specifications for a bill of materials) and publicly available data from outside sources (such as commodity prices or information on product substitutes or complements from competitors).

The important point to remember for

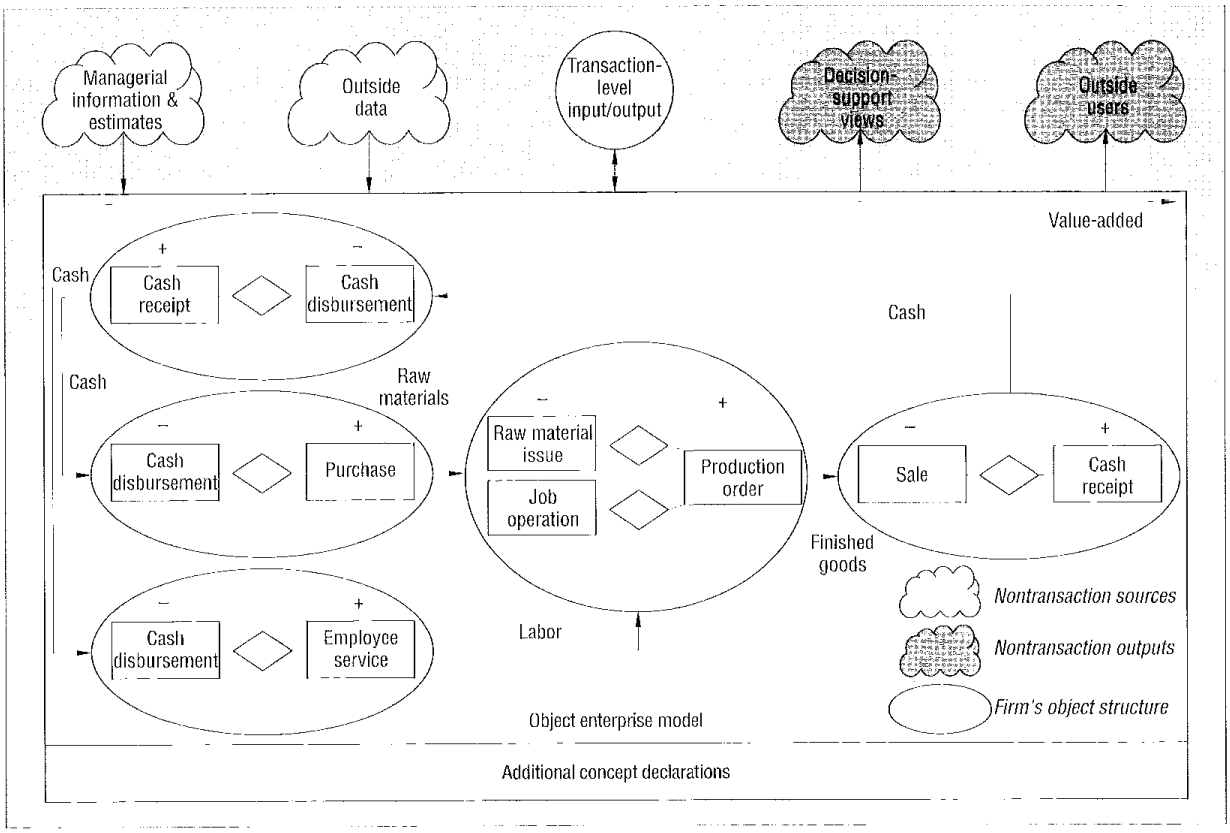

Figure 2. REA object infrastructure for an enterprise.

enterprise operation is that REA specification of economic phenomena allows semantic integration of data from disparate sources. Thus a piece of inventory could be given an integrated description of its cost and availability (from input transaction data), physical specifications (from engineering estimates), and competitiveness (from outside data sources). Such integrated semantics are impossible in traditional business information systems that rely on bookkeeping or data-processing artifacts for object-classification purposes.

Figure 2 also shows the nontransaction outputs associated with operating the REA object enterprise model daily. The coupling with decision-support systems—especially if those systems contain semantically specified components—might change dramatically, as an REA enterprise information architecture maintains its object- and process-level semantics within itself. In a way, its meaning is conveyed with its data, and this makes any connection to another intelligent system less problematic because it leaves less room for misinterpretation. This "conveyed meaning" also makes automated use of the enterprise model's components easier for users outside the firm. Such use would certainly facilitate the development of electronic commerce where two firms exchange data based on common semantic specifications rather than on enforced adherence to a syntactic EDI document standard.

An ontological component of these systems, which perhaps differentiates them the most from traditional accounting and data-processing architectures, is the specification of additional concepts, which can be derived logically from the base declarations (see Figure 2).[9] Repeatedly using a standard object template to construct an REA enterprise model allows automated reasoning (involving specific pattern matches on all appropriate object constellations) to occur at the highest possible concept-definition level. Instead of having to write multiple procedures to define various types of economic derivatives, such as claims (which in REA terms are imbalances between sets of increments and decrements), we find that we can define such a concept once and let a reasoner find its instantiations. This makes the semantics of such definitions much clearer by removing them from procedures and making them declarative.[10]

Coupling with knowledge-based decision tools

There have been only a handful of directed REA implementations in actual companies such as Alcoa and Sears. However, firms such as PricewaterhouseCoopers and IBM have adopted some REA standards as guiding architectural features for accounting system design,[2] and SES Software has adopted many of its principles for business-object design in its BOMA architecture.[11] There are also firms explicitly using REA to develop software for niche markets plagued by accounting system integration problems, such as supply-chain synchronization with intelligent agent technology.[12]

None of these implementations, however, have been *full REA* models in the sense of the object enterprise model Figure 2 shows, which specifies all processes and objects, implementing them without cost-benefit compromise. With the advent of enterprise resource planning systems and the development of business object architectures, the software marketplace has increasingly moved toward implementing value-chain-oriented (that is, REA-like) accounting infrastructures. Nonetheless, this movement

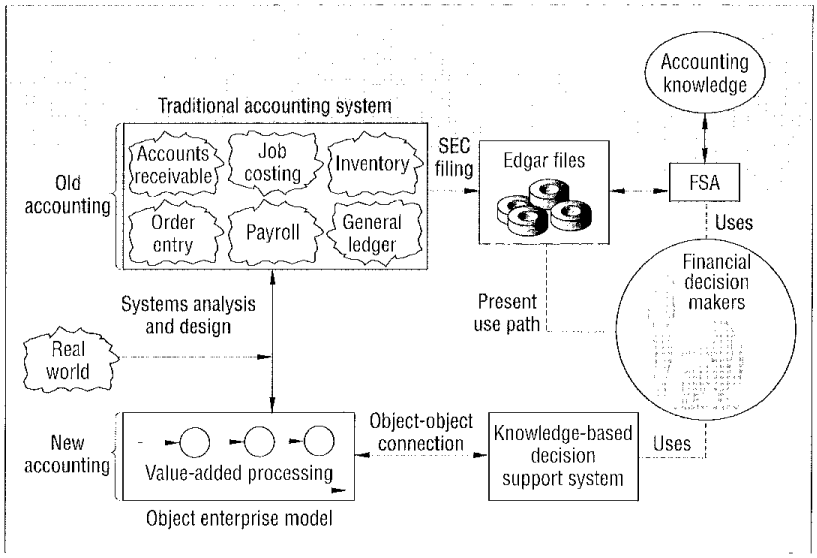

Figure 3. Intelligent system use with different accounting systems.

away from the legacy world is gradual and, in many cases, it is still subtly characterized by implementation platforms whose explicit object semantics and entrepreneur rationale are partially obscured by overreliance on conventional accounting conventions and general-ledger-oriented thinking.

Here, however, we assume we could overcome the organizational constraints of entrenched legacy solutions and the technology constraints of processing time and storage capacities, and we speculate in Figure 3 how such full-process models might be linked with certain types of knowledge-based systems. This assumption and its accompanying discussion is quite realistic in a design environment where the full possibilities for an enterprise architecture are considered thoroughly in an early assessment phase unfettered by cost and technology constraints.

Figure 3 illustrates how information about the real world might be filtered through to financial decision makers through two different types of accounting systems. Both the old (general-ledger based) and new (REA-based) accounting systems show proposed use of knowledge-based technology with dotted lines emanating from the decision makers; the difference is in the nature of their linkages to the enterprise's actual information system.

The upper half of Figure 3 portrays a conventional accounting environment, and it also illustrates the architecture needed for FSA (financial statement analyzer),[13] one of the early (1987) AI prototypes intended to be used with the Edgar (Electronic Data Gathering, Analysis, and Retrieval) system. Edgar is an information gathering and dissemination system run by the Securities and Exchange Commission (SEC) of the US government, and it is intended to make financial data on publicly traded corporations available to individuals and institutions in the US capital markets. Edgar has enjoyed an enormous upswing in Internet use since its public introduction in the mid 1990s, although its output is difficult to interpret and understand unless you are an expert at financial-statement analysis and comparison. What the Edgar system offers the investing public (its *present use path* in the diagram) is little more than a word-processing text of company filings with the SEC (although there certainly have been efforts to upgrade the quality of this interface with knowledge-based tools[14]).

The intelligent system FSA was able to take automated corporate filing data (from an SEC 10-K report, for example) and calculate financial ratios that analysts commonly use in assessing a corporation's financial well-being. This seems like a relatively straightforward task—indeed, it is often taught in undergraduate finance and accounting classes. Having a machine handle the task completely, however, caused the FSA designers from Arthur Andersen to confront the highly idiosyncratic nature of old accounting systems.

First, an exhaustive chart-of-accounts knowledge structure had to be built into FSA because of the individualistic and synonymous naming conventions used by many companies to label various asset, liability, equity, income, and expense accounts. Second, because some of the actual account information is buried in textual footnotes, FSA also had to be equipped with natural-language-processing capabilities for certain limited cases such as the contra-accounts for depreciation (on inventory) and subleases (on rental expense). These are adjustments that expert human analysts find somewhat easy, but that cause significant interpretation problems for a fully automated system.

FSA worked well in its very limited domain, although it did require a high level of expertise with knowledge representation structures and tradeoffs, with knowledge-acquisition problems, and with object-oriented programming techniques to make it operational. A companion AI system called Eloise (used for natural-language processing of other 10-K material) was also built under contract with the SEC for Edgar by Arthur Andersen at approximately the same time. However, neither of these systems nor any similar AI efforts of the late 1980s were part of the production versions of Edgar that were implemented in the 1990s. Therefore, the only widely available option for prospective users is the indicated direct link to the Edgar files.

The SEC did not make public their rationale for this exclusion, but a compelling case can be made for one noncost reason why they didn't attempt it. All the semantics necessary for a system such as FSA to function had to come from the AI tool itself because of the idiosyncratic and syntactic nature of the accounting reporting systems. Good examples of such idiosyncrasies were the many conventions needed to cover receivables.[13]

As we mentioned, REA coverage of such claims is much more direct and declarative. If the accounting systems in question had more of a semantic base, building intelligent systems to work with them might not have been such a daunting hurdle. Currently, Edgar's disseminated output contains lots of data but very little assistance in determining the meaning of that data. Therefore, it is no surprise that more sophisticated but expensive alternatives exist in the marketplace to process and restructure Edgar output for users who find its present offerings less than usable. Quite simply, the present Edgar system cannot be used in any knowledge-intensive way.

Figure 3 also illustrates how a reconfigured decision-support system such as FSA might work for financial decision makers

in an REA environment. The process and (more importantly in this case) the object semantics in such an implementation remain intact and reside with the enterprise model. Unlike FSA, which had to be augmented with account-hierarchy and footnote-schemata knowledge structures, our new knowledge-based system would need only the structures associated with the specific decision expertise (such as how to value proprietary resources or whether to invest in a certain type of stock). If this expertise were coded as a semantic network with the same ontological commitment to REA-based representation, the coupling between the two systems would be especially close. Essentially, the intelligent system would specify the concepts only at the type level, with individual consultations being instantiated with direct object-object connections.

The organizational and capital-market ramifications of such direct inside-database to outside-decision-maker links involve factors and changes that numerous accounting theorists and practitioners have described—most recently SEC Commissioner Steven Wallman.[15] A chief issue is the idea that the general investing public should have the same access to public financial data as analysts with extended rights to knowledge-laden processing. Even though there are numerous technological hurdles (in addition to the semantic incompatibility problems we discussed) to such direct disclosure links, there is no doubt in the minds of people such as Wallman about their ultimate desirability.

Knowledge-intensive enterprise-systems design

While it is certainly true that our models look quite different from traditional accounting architectures built on bookkeeping ideas, it is also the case that there are enterprise software packages that afford hospitable implementation platforms for systems built on REA principles. This is because many of the database tenets on which REA was originally based in 1982 are features that make business software solutions attractive in the late 1990s—an emphasis on strong semantics, versatile use and delayed procedural aggregation of economic transaction data, and a strong orientation toward wider communities of users to include accountants and nonaccountants. Additionally, more recent enterprisewide modeling extensions[3] add value-chain and workflow infrastructures above and below that of the REA-patterned process level in an integrated fashion, and these types of additions are the principle features of many high-end software solutions for enterprises.

We think it's possible to take a strongly directed REA approach to building enterprise models that can serve both as blueprints for strategic information architectures and as initial database schemas that can be compromised by cost-benefit considerations in individual companies. In this sense, our frameworks are like the prototypical models for certain lines of businesses that some analysis methodologies[7] advocate as starting points for information-system design.

Our research and practical implementation work with REA modeling of accounting phenomena has progressed on a number of software-engineering and empirical-validation fronts.[16] For example, we built the two following knowledge-based systems with these principles. The first is Reach,[17] a CASE (computer-aided software engineering) tool for view modeling and integration that uses three different types of knowledge: *first-order principles* of the REA template, *heuristic* guidance of implementation compromises based on object pattern matches, and *reconstructive* expertise for prototypical models based on library guides for designing account-based bookkeeping systems.

The second system is Creasy.[10] Also a CASE tool, it supports conceptual and operational design of full REA models. The Creasy environment embeds both methods knowledge (of semantic modeling structures and constraints) and domain-specific knowledge (of REA accounting) for automated use by novice modelers and users. We review in an earlier work these and other REA tools (such as the REAtool schema evolution system[18]) in an integrated methodological fashion[19] and suggest other possible instances where REA-enabled knowledge can assist an enterprise modeler with information-system analysis, design, and implementation.

Research extensions and implementation work

The software engineering research we describe constitutes only one of the directions where we feel design work with semantic accounting models should be headed. Two other important areas are integrating REA models with research in the area of enterprise ontologies and extending REA implementation work in the direction of design patterns.

Ontological directions. An ontology, according to Tom Gruber,[20] is an explicit specification of the entities (objects and concepts) and the relationships that hold among them in an abstract and simplified view of the world that is represented for some computational purpose. Certainly, the REA model shown in Figure 1 qualifies conceptually as an accounting ontology. It is a specification of the set of objects and the describable relationships among them and their mediating purpose[9] that exists most narrowly in an accounting universe of discourse and most broadly as the accountability infrastructure for an enterprise universe of discourse.

More importantly, the REA ontological framework is a semantic specification of a conceptualization that has been published and peer-reviewed in its home discipline.[1] Furthermore, it is a pedagogical framework of entities, relationships, and economic purpose that is used widely for instruction in the educational programs of its home discipline.[6] These are two criteria to which most ontologies should aspire but don't. The journal source validates the object definitions and their classification hierarchies. The textbook employment insures that the model's ideas are useful in understanding and interpreting the phenomena that the ontology purports to embody.

However, despite these advantages, we must extend REA's ontological features if we are going to use it comprehensively for enterprise knowledge management in the fashion Dan O'Leary describes.[21] REA's theoretical features need amplification and more grounding to supra-accounting theories in strategic management,[5] so that we can use them for both accountability and policy-making purposes. They also need ontological conceptual analysis as Natalya Noy and Carole Hafner[21] suggested for corresponding business features such as the explicit treatment of time.

Additionally, all of the individual cycle instances of the model's entities, relationships, processes, tasks, and so forth must be cataloged and made available for automated use with specification in an ontology definition language such as Ontolingua,[22] and its limited inventory[10] of logical axioms must

be expanded considerably. And finally, because REA is actually a compact domain-specific ontology for transaction-oriented economic phenomena, it must be expanded and integrated with concepts from other enterprise ontologies,[7] concepts from nonaccounting domains such as supply-chain management and workflow management, and with more general-purpose ontologies.[22] Hopefully, interested AI practitioners who have found the prospect of building intelligent systems that interact with artifact-laden accounting systems somewhat difficult will perform this REA extension work.

Implementation directions. Research work such as REA becomes much more useful (and reproducible) to a software-design community if it is promulgated in explicit pattern form. Such an orientation is the focus of our present object-oriented implementation work at Michigan State. We are trying in our Reaper[23] project to specify and catalog the patterned behavior needed to model the semantics of the various relationships shown in Figure 1b. For example, the duality relationship can specify matching between the give events and take events in a multitude of ways (with declarative links between single or aggregate transactions, with procedural links, and so forth). The knowledge-level logic of these various matching schemes often parallels conventional accounting schemes such as period expensing or activity-based costing. By codifying these behaviors in pattern form, we hope to make REA solutions more widely available and reusable for designers trying to upgrade from legacy accounting systems to more knowledge-based environments. ■

References

1. W.E. McCarthy, "The REA Accounting Model: A Generalized Framework for Accounting Systems in a Shared Data Environment," *The Accounting Rev.*, Vol. 57, No. 3, July 1982, pp. 554–78.
2. K.B. Walker and E. Denna, "A New Accounting System Is Emerging" *Management Accounting*, July 1997, pp. 22–30.
3. G.L. Geerts and W.E. McCarthy, "Modeling Business Enterprises as Value-Added Process Hierarchies with Resource-Event-Agent Object Templates," *Business Object Design and Implementation*, J. Sutherland and D. Patel, eds., Springer-Verlag, New York, 1997, pp. 94–113.
4. G.L. Geerts and W.E. McCarthy, "The Economic and Strategic Structure of REA Accounting Systems," 1994; http://www.reavillage.org (July 1999).
5. M.E. Porter, "Towards a Dynamic Theory of Strategy," *Strategic Management J.*, Vol. 12, 1991, pp. 95–117.
6. A.S. Hollander, E.L. Denna, and J.O.Cherrington, *Accounting, Information Technology and Business Solutions*, Irwin/McGraw Hill, Burr Ridge, Ill., 1995.
7. P. Bernus and L. Nemes, eds., *Modelling and Methodologies for Enterprise Integration*, Chapman and Hall, London, 1996.
8. G.L. Geerts and W.E. McCarthy, "Using Object Templates from the REA Accounting Model to Engineer Business Processes and Tasks," 1997; http://www.reavillage.org (July 1999).
9. J. Sowa, *Knowledge Representation: Logical, Philosophical, and Computational Foundations*, to be published, PWS Publishing Company, Aug. 1999.
10. G.L. Geerts and W.E. McCarthy, "The Extended Use of Intensional Reasoning and Epistemologically Adequate Representations in Knowledge-Based Accounting Systems," *Proc. 12th Int'l Workshop on Expert Systems and Their Applications*, 1992, pp. 321–332.
11. C. Marshall, "Business Object Management Architecture," 1998; http://www.jeffsutherland.com/oopsla97/marshall.html (July 1999).
12. R. Haugen, "Radically Distributed Supply Chain Systems," 1998; http://www.jeffsutherland.com/oopsla97/haugen.html (July 1999).
13. C. Mui and W.E. McCarthy, "FSA: Applying AI Techniques to the Familiarization Phase of Financial Decision Making," *IEEE Expert*, Vol. 2, No. 3, Fall 1987, pp. 33–41.
14. K.M. Nelson et al., "Virtual Auditing Agents: The EDGAR Agent Challenge," to be published in *Decision Support Systems*, Elsevier Science, New York, 1999.
15. S.M.H. Wallman, "The Future of Accounting and Financial Reporting, Part IV: 'Access' Accounting," *Accounting Horizons*, Vol. 11, No. 2, 1997, pp. 103–116.
16. C.L. Dunn and W.E. McCarthy, "The REA Accounting Model: Intellectual Heritage and Prospects for Progress," *J. Information Systems*, Vol. 11, No. 1, Spring 1997, pp. 31–51.
17. S. Rockwell and W.E. McCarthy, "REACH: Automated Database Design Integrating First-Order Theories, Reconstructive Expertise, and Implementation Heuristics for Accounting Information Systems," to be published, *Int'l J. Intelligent Systems in Accounting, Management and Finance*, John Wiley and Sons, Ltd., Sept. 1999.
18. N. Chen, D. O'Leary, and D. McLeod, "Schema Evolution for Object-Based Accounting Database Systems," *Expert Systems with Applications*, Vol. 9, No. 4, 1995, pp. 491–502.
19. G.L. Geerts, W.E. McCarthy, and S.R. Rockwell, "Automated Integration of Enterprise Accounting Models throughout the Systems Development Life Cycle," *Int'l J. Intelligent Systems in Accounting, Management and Finance*, Vol. 5, Sept. 1996, pp. 113–128.
20. T. Gruber, "A Translational Approach to Portable Ontologies," *Knowledge Acquisition*, Vol. 5, No. 2, 1993, pp. 199–220.
21. D. O'Leary, "Using AI in Knowledge Management: Knowledge Bases and Ontologies" *IEEE Intelligent Systems*, Vol. 3, No. 3, May/June 1998, pp. 34–39.
22. N.F. Noy and C.D. Hafner, "The State of the Art in Ontology Design: A Comparative Review," *AI Magazine*, Vol. 18, No. 3, Fall 1997, pp. 53–74.
23. G.L. Geerts and W.E. McCarthy, "Accounting as Romance: Patterns of Unrequited Love and Incomplete Exchanges in Life and in Business Software," as summarized by J. Sutherland, *OOPSLA 97 Addendum*, ACM Press, New York, 1998.

Guido L. Geerts is an assistant professor of accounting and information systems at Michigan State Univeristy. His research interests include semantic modeling of accounting phenomena, enterprise ontologies, knowledge-based systems, object technologies, and Internet technologies. He received his PhD in economics from the Free University of Brussels. He is a member of the IEEE and American Accounting Association. Contact him at the Dept. of Accounting, N259 N. Business Complex, Michigan State Univ., E. Lansing, Minn. 48824; geerts@pilot.msu.edu.

William E. McCarthy is the Arthur Andersen Professor of accounting and information systems at Michigan State University. He is also the chair of the Artificial Intelligence & Emerging Technologies Section of the American Accounting Association. His research interests include semantic modeling of accounting phenomena, domain-specific ontology development, enterprise modeling, and knowledge-based systems. He received his PhD in accounting and computer science from the University of Massachusetts at Amherst. Contact him at the Dept. of Accounting, N270 N. Business Complex, Michigan State Univ., E. Lansing, Minn. 48824; mccarth4@pilot.msu.edu.